UNIVERSIDAD AUTÓNOMA DE YUCATÁN
FACULTAD DE MATEMÁTICAS

SISTEMA DE MONITOREO Y EMISOR DE RECOMENDACIONES BASADO EN APRENDIZAJE AUTOMÁTICO Y ONTOLOGÍAS

TESIS

QUE PARA OBTENER EL GRADO DE

Maestro en Ciencias de la Computación

SUPERVISORES

DR. JORGE RICARDO GÓMEZ MONTALVO

DR. JESÚS IXBALANK TORRES ZÚÑIGA

PRESENTA

CHRISTIAN JESÚS VADILLO MEJÍA

Noviembre 2020

Dedicatoria

Dedico mi trabajo de tesis a mi familia y amigos que me acompañaron durante esta aventura. Una dedicación especial hacia mis amados padres, Margarita Mejía y Manuel Vadillo, por el apoyo incondicional durante cada una de las fases vividas en esta etapa de mi vida. A mis hermanos, Paola y Manuel que siempre estuvieron pendientes de mis pasos.

Agradecimientos

Agradezco al Consejo de Ciencia y Tecnología (CONACYT) el apoyo otorgado a través de la Beca para Estudios de Maestría No. 922808. Asimismo, agradezco al Dr. Jorge R. Gómez Montalvo y al Dr. Jesús I. Torres Zúñiga por su apoyo y orientación en este trabajo de titulación.

Resumen

Uno de los desafíos en la operación de procesos industriales es la seguridad y confiabilidad del sistema. Este punto es importante ya que se manejan recursos potencialmente peligrosos para el entorno natural. Además, se sabe que todo sistema construido por humanos es poco confiable y propenso a fallas. Por ejemplo, debido a la degradación con el tiempo y/o el uso se producen fallas en los procesos del sistema.

En años recientes, el aprendizaje automático se ha empleado para labores de detección de fallas con resultados favorables [1–7]. En un modelo de aprendizaje automático, los datos recopilados son utilizados para el entrenamiento del modelo con el fin de encontrar patrones ocultos en los datos producidos por el proceso de interés.

Por otro lado, uno de las desventajas principales de los modelos de aprendizaje automático es la pobre interpretación que se le puede dar a sus resultados. Por ejemplo, un modelo de aprendizaje automático que detecta (con cierto nivel de fiabilidad) si un proceso específico presenta un problema. De nuevo, la única información de referencia obtenida es 0 o 1. El problema surge cuando el operador o experto desea localizar la falla, teniendo nula información de referencia y haciendo más lenta su solución.

Para dar solución a este problema, en este trabajo se plantea un enfoque para un sistema de monitoreo y detección de fallas que es aplicado a una biorrefinería de producción de hidrógeno. El sistema de monitoreo y detección de fallas está basado en un modelo de aprendizaje automático y un modelo ontológico. Por un lado, se realiza la detección de fallas utilizando un modelo Random Forest, el cual determina si un proceso esta funcionando correctamente o si existe un problema. Posteriormente, el modelo ontológico se encarga de inferir con mas detalle el estado de los procesos, las fallas o posibles fallas y el alcance de estas. Esto se hace a través de la modelación de entidades y relaciones extraídas del conocimiento del experto.

Los resultados mostraron que el uso del modelo ontológico no sólo facilita el diagnóstico de fallas en los procesos, sino que es capaz de detectar fallas que el modelo de aprendizaje automático fue incapaz de detectar.

Por último, se presenta una plataforma web que encapsula cada uno de los componentes descritos previamente. El sistema de monitoreo web es capaz de registrar mediciones nuevas, detectar fallas y emitir recomendaciones.

Tabla de Contenido

Índice de tablas

Índice de figuras

Abreviaciones

AGV	Ácidos Grasos Volátiles
AI	Artificial Intelligence
AL	Attributive Language
ANN	Artificial Neural Network
API	Application Programming Interface
CBR	Case-Based Reasoning
CNN	Convolutional Neural Network
DA	Digestor Anaerobio
DL	Description Language
DQO	Demanda Química de Oxígeno
FBR	Fotobioreactor
FOL	First-Order Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
KB	Knowledge Base
KIF	Knowledge Interchange Format
KR	Knowledge Representation
MEC	Microbial Electrolysis Cell
MVA	Multivariate Analysis
ODE	Ordinary Differential Equation
OIL	Ontology Inference Layer
OWL	Web Ontology Language
PCA	Principal Component Analysis
POO	Programación Orientada a Objetos

RBR	Rule-Based Reasoning
RDFS	Resource Description Framework Schema
RF	Random Forest
SQL	Structured Query Language
SVM	Support Vector Machines
SWRL	Semantic Web Rule Language
URL	Uniform Resource Locator
XML	Extensible Markup Language
XOL	Ontology Exchange Language

Capítulo 1

Introducción

1.1 Preliminares

El significante aumento en las emisiones de CO_2 [9] en años recientes, ha generado una necesidad urgente de desarrollar alternativas a la industria basada en recursos fósiles. Una opción prometedora para cubrir esta tarea, son las biorrefinerías y la biomasa, cuyo término hace referencia a la materia orgánica renovable, como árboles, cultivos agrícolas, algas y varios residuos o desperdicios [10].

Uno de los desafíos presentes en las biorrefinerías, es la capacidad de sustentabilidad [11]. Básicamente, lograr que una biorrefinería sea más sustentable es hacer que evolucione de manera equilibrada y simultáneamente en tres dimensiones: economía, social y medioambiental [11]. Este último punto es particularmente crítico, ya que las biorrefinerías se centran en la utilización de residuos orgánicos para la producción de energía.

Así, uno de los problemas comunes en el diseño de biorrefinerías es la optimización de las condiciones de operación con el fin de hacer un uso más eficiente de los residuos y maximizar la productividad del proceso [10]. Sin embargo, la naturaleza compleja de una biorrefinería, la cual se caracteriza por manejar una gran cantidad de variables de estado, las cuales generan una gran cantidad (miles o millones) de datos [10], plantean una serie de desafíos para operar en condiciones óptimas una biorrefinería.

En general, uno de los problemas más críticos, es la seguridad y confiabilidad del proceso. La ingeniería del control, una disciplina que se ocupa de los fundamentos formales del análisis y diseño de los sistemas de control y gestión de sistemas [12], establece que todo sistema construido por humanos es poco confiable [13]. La degradación con el tiempo y/o el uso de cada uno de los componentes de una biorrefinería (sensores, actuadores, tuberías, reactores, medios biológicos, etc.) producen fallas en los procesos del sistema. Algunas fallas pueden no tener un gran impacto, por ejemplo, una disminución en la producción, pero otras pueden causar pérdidas millonarias de recursos y graves daños al medio ambiente [14–17].

Una forma común de afrontar estos problemas es mejorar la calidad, confiabilidad y robustez de los componentes individuales, como sensores, actuadores, reactores, instalaciones o computadoras. Sin embargo, aún mejorando cada componente individual, no es posible

garantizar un sistema totalmente libre de fallas funcionales [18].

Así, a medida que los sistemas industriales modernos crecen y se vuelven complejos, el uso oportuno de sistemas de monitoreo y diagnóstico de fallas se hace más relevante [19]. Estos nos permiten garantizar la seguridad de los procesos en los sistemas [2], al monitorear los niveles óptimos definidos; facilitar la toma de decisiones [20], al tener una mejor comprensión del comportamiento del sistema industrial; y mejorar la calidad del producto industrial [21], al tener funcionando en optimas condiciones los procesos del sistema. Por lo tanto, dada la incapacidad de tener un sistema totalmente seguro y confiable, los sistemas de monitoreo de procesos y detección de anomalía han captado la atención tanto del sector académico e industrial en los últimos años [22–30], sin embargo, muchos de los enfoques carecen de una interpretación a nivel humano que facilite la toma de decisiones.

1.2 Contexto y problématica

Desde el punto de vista de ingeniería de control, la complejidad biológica asociada a los procesos de una biorrefinería resulta en sistemas dinámicos descritos por modelos altamente no lineales, con incertidumbres paramétricas, propensos a múltiples perturbaciones externas y anomalías. Además, los procesos que los constituyen suelen operar con un gran número de variables, y debido a su naturaleza altamente correlacionada, la presencia de una anomalía en un proceso específico puede influir en otros a través de la propagación de la misma [22], provocando un aumento en el tiempo de inactividad del sistema.

De forma general, una anomalía o un valor atípico se define como una observación que es inconsistente con el comportamiento normal de un elemento [31]. A menudo, suelen ser más interesantes y contener información más valiosa que las mediciones de estados normales. Por ello la importancia de desarrollar esquemas de control y monitoreo robustos ante las condiciones descritas previamente, considerando también, las limitaciones en cuanto al número y calidad de las mediciones recolectadas de los procesos y de los modelos matemáticos de los que se dispone [32].

En años recientes, el aprendizaje automático se ha empleado en la detección de anomalías con resultados favorables [1–7]. En un modelo de aprendizaje automático, los datos recopilados son utilizados para el entrenamiento del modelo con el fin de encontrar patrones ocultos en los datos de los procesos de un sistema.

Por otro lado, uno de las desventajas principales de los modelos de aprendizaje automático es la pobre interpretación que se le puede dar a sus resultados. Por ejemplo, ¿confiaría ciegamente en el resultado de un modelo que predice si padece o no de cáncer?. El único resultado que un médico o un paciente sería capaz de ver es un 0 o un 1 (ausencia o presencia), sin ningúna otra información extra que sustente esta decisión. Otro ejemplo sería un modelo de aprendizaje automático que detecta (con cierto nivel de fiabilidad) si un proceso específico está presentando un problema. De nuevo, la única información de referencia obtenida es 0 o 1. El problema surge a la hora en que el ingeniero desea localizar la falla, teniendo nula información de referencia y haciendo más lenta su solución.

Por lo tanto, identificar una anomalía utilizando un modelo de aprendizaje automático

solo es la mitad del problema, es necesario poder identificar tanto la gravedad de la anomalía, como las variables involucradas. De esta forma se puede realizar un diagnóstico más certero del mundo modelado y poder emitir una sugerencia o recomendación.

Los desafíos planteados previamente generan las siguientes interrogantes, que en nuestro caso se plantean para una biorrefinería de producción de hidrógeno.

1.2.1 ¿Cómo modelar un sistema de monitoreo para una biorrefinería?

En la literatura podemos encontrar varios métodos para la creación de modelos de monitoreo. De forma general, hay dos que resaltan: las técnicas de monitoreo basados en modelados clásicos y las basadas en métodos de *data-driven*. La primera es más robusta pero difícil de realizar si no se cuenta con un amplio conocimiento del sistema. Por otra parte, las técnicas basadas en *data-driven* construyen modelos completamente a partir de los datos de los procesos. Estas técnicas han mostrado ser más efectivas en la práctica [26, 33–35]. Dado que, los procesos de la biorrefinería rutinariamente generar cientos de datos en poco tiempo, se cuenta con demasiada información histórica de los procesos. En este caso, el uso de técnicas de *data-driven* es una opción a considerar para este sistema. Entre las preguntas que surgen de este problema se encuentran:

- ¿Qué técnica brinda la mejor solución al contexto presentado?
- ¿Se tiene información de calidad para el modelado del sistema?

1.2.2 ¿Cómo identificar una anomalía en el contexto presentado?

Un aspecto importante de la detección de anomalías es la naturaleza de la misma. En [36] se menciona la siguiente clasificación:

- Anomalía puntual: Cuando una instancia de datos particular se desvía del patrón normal del conjunto de datos.
- Anomalía contextual: Cuando una instancia de datos se comporta de manera anómala en un contexto particular.
- Anomalía colectiva: Cuando una colección de instancias de datos similares se comporta de forma anómala con respecto a todo el conjunto de datos.

Cabe destacar que, dada la naturaleza compleja de la biorrefinería, detectar una anomalía implica determinar el alcance de esta. Es decir, ¿Podemos determinar la influencia que tiene una anomalía en un proceso especifico hacia otros procesos?

Entre las preguntas que surgen de este problema se encuentran:

- ¿Cómo identificar una anomalía?
- ¿Cómo reconocer su naturaleza?

- ¿Cómo determinar el alcance de la anomalía?
- ¿Es posible detectar múltiples anomalías de forma simultánea?

1.2.3 ¿Es posible emitir recomendaciones acorde al estado de la biorrefinería?

Una de las finalidades principales de un sistema de monitoreo es la toma de decisiones basadas en cierta información sobre cierto proceso monitoreado [12]. Para ello se debe tener un fácil acceso a la información necesaria para la toma de decisiones. Entre las preguntas que surgen de este problema se encuentran:

- ¿Cómo facilitar el acceso a la información del sistema de monitoreo?
- ¿Cómo generar una recomendación con base a un estado?
- ¿Cómo presentar la información para facilitar la toma de decisiones?
- ¿Puede el sistema de monitoreo trabajar en tiempo real y en línea?

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un sistema de monitoreo usando técnicas de aprendizaje automático que detecte anomalías, sea interpretable y genere recomendaciones para los operadores de una biorrefinería de producción de hidrógeno.

1.3.2 Objetivos específicos

- Determinar los límites y riesgos latentes para cada proceso.
- Desarrollar un modelo para la detección de un estado normal y anormal.
- Crear un sistema de diagnóstico y emisión de recomendaciones.
- Implementar el sistema propuesto en un contexto web.

1.4 Contribuciones

En este proyecto de tesis se desarrolló un sistema de monitoreo que detecta anomalías y emite recomendaciones (SMDAER) para los operadores de una biorrefinería de producción de hidrógeno.

Se ha usado una combinación de dos modelos para la detección de anomalías. Primero, se han entrenado dos clasificadores *Random Forest* para diagnosticar, de forma general, el

estado de la biorrefinería. Posteriormente, el modelo ontológico reafirma este primer diagnóstico y genera recomendaciones de ser necesario. Sin embargo, se sabe que ningún sistema es totalmente confiable, por eso se le otorga al operador la capacidad de aprobar o rechazar el diagnóstico emitido. Esto sirve de retroalimentación para el futuro re-entrenamiento del clasificador.

Por otra parte, el modelo ontológico que se presenta evita al operador el tener que manipular directamente el código del sistema de monitoreo para agregar o modificar errores, variables o procesos. Esto facilita la calibración de los límites para la clasificación de estados a través de los modelos clasificadores. Además, el enfoque presentado puede ser extrapolado a otros dominios similares. De igual forma, el clasificador *Random Forest* puede ser sustituido sin afectar al sistema de monitoreo.

Cabe recalcar que la plataforma propuesta en este proyecto de tesis fue una prueba de concepto (proof of concept en inglés), por las siguientes razones:

- Las pruebas experimentales para evaluar el funcionamiento de la plataforma no fueron en un entorno real.
- La biorrefinería descrita en este documento aún está en estado de propuesta. Es decir, físicamente aún no está en funcionamiento.
- Los datos provienen de un simulador facilitado por el doctorante José de Jesús Colín Roblesla de la Universidad de Guanajuato.

Se espera que el sistema sea introducido como una herramienta de apoyo para la toma de decisiones en la operación de la biorrefinería propuesta.

1.5 Estructura de la tesis

Esta tesis se encuentra estructurada de la siguiente forma:

En el Capítulo 2 se introduce la arquitectura general de la plataforma SMDAER. Además, se presenta la metodología seguida para la generación de datos anómalos.

En el Capítulo 3, se presenta un modelo de clasificación para detección de anomalías utilizando un clasificador *Random Forest* y los datos generados en el Capítulo 2

En el Capítulo 4 se describe la metodología y el diseño del modelo ontológico de la biorrefinería. De igual forma, se presenta un modelo de clasificación de estados utilizando la ontología propuesta y reglas SWRL. También se describe el modelo de emisión de recomendaciones con base al modelo ontológico propuesto.

En el Capítulo 5 se describe la arquitectura y el diseño de un sistema web usado para la interacción del operador con los modelos creados. Se presentan los experimentos realizados para medir la detección de anomalías, emisión de recomendaciones y el análisis de los resultados.

Finalmente, en el Capítulo 6 se dan las conclusiones del proyecto, así como la propuesta de trabajo futuro.

Capítulo 2

Simulador de una Biorrefinería y arquitectura general del sistema de monitoreo

2.1 Biorrefinería de producción de hidrógeno

Particularmente, la biorrefinería con la que se trabajó utiliza efluentes agroindustriales como materia prima. Existen diferentes tipos de tratamiento para efluentes agroindustriales que incluyen tratamientos químicos, físicos, fisicoquímicos y biológicos. Sin embargo, es bien sabido que los tratamientos biológicos son los más rentables y eficientes en términos de insumos, tiempos de tratamiento y desempeño [32]. Así, también es bien conocido que el único proceso biológico capaz de tratar directamente (sin dilución) efluentes agroindustriales con Demanda Química de Oxígeno (DQO) superior a 20 $gDCO/l$ es la digestión anaerobia (DA) [37]. Desde un punto de vista de tratamiento biológico, una ventaja de emplear biorreactores anaerobios para el tratamiento de efluentes orgánicos es la producción de biogás que incluye CO_2, CH_4 y $bioH2$ [37].

Con respecto a $bioH2$ también se han propuesto recientemente las Celdas Electrolíticas Microbianas (MEC por sus siglas en inglés) como una alternativa económica, eficiente y prometedora para su producción, en las que un tipo de bacterias referidas como exoelectrógenas oxidan un sustrato y liberan electrones en el ánodo [38, 39]. Los electrónes viajan desde el ánodo al cátodo a través de un conductor eléctrico y en la cámara catódica se produce el bioH2 por medio de una electrólisis.

Por otro parte, el aprovechamiento de CO_2 proveniente de procesos como la DA para el crecimiento autótrofo de microalgas también ha sido ampliamente estudiado recientemente para la purificación del biogás producido en digestores anaerobios y la producción de biomasa [40, 41]. Así, el biogás producido queda libre de CO_2, mientras que la biomasa microalgal puede ser aprovechada directamente como biofertilizante, o bien para la producción de pigmentos, o biocombustibles como biodiesel [32]. Además, los ácidos grasos volátiles (AGV)

provenientes de la primera fase de la DA (acidogénesis bajo condiciones anaerobias, AcDA) son adecuados para ser utilizados como afluentes de los sistemas bioelectroquímicos [32], mientras que el CO_2 producido puede ser usado en la producción de microalgas (PBR por sus siglas en inglés).

Así, visto como una biorrefinería, la combinación de estos procesos puede resultar en una explotación potencialmente completa de flujos de residuos orgánicos, que maximiza al mismo tiempo la recuperación de energía y la descontaminación de efluentes.

2.2 Estructura básica del simulador de datos

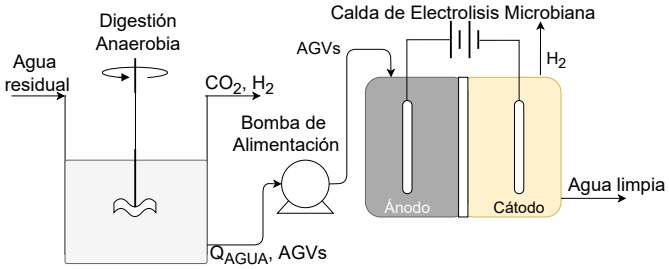

Figura 2.1: DA acoplado a una MEC

En la sección previa, introducimos la propuesta de una biorrefinería de producción de hidrógeno. Esta propuesta contempla el acoplamiento de tres procesos principales, la digestión anaerobia, las celdas de electrólisis microbianas y un fotobiorreactor. Por su carácter de propuesta, aún se requiere validar los modelos de operación para los tres tipos de reactores descritos anteriormente. Ya sea por separado u operando de manera acoplada. Esta situación conlleva a no contar con datos experimentalesdel funcionamiento de esta biorrefinería. Sin embargo, un equipo de la Universidad de Guadalajara y de la Universidad de Guanajuato ha desarrollado un simulador [42] del digestor anaerobio acoplado a una celda de electrolisis microbiana (Figura 2.1). Este simulador es capaz de emular un funcionamiento correcto de los primeros dos procesos propuestos, DA y MEC. El proceso de digestión anaerobia está acoplado a la entrada de una celda de electrólisis microbiana utilizando una concentración de materia orgánica en el agua residual que varía con el tiempo de forma senoidal. El simulador puede generar datos para un periodo aproximado de 10 años y es el que se ha usado como referencia para el desarrollo de este proyecto de investigación. De esta manera, en este trabajo estaremos enfocados en el proceso de DA acoplado a la MEC. El monitoreo y detección de anomalías del PBR se propondrá como un trabajo a futuro, pues el simulador y el proceso están en desarrollo.

El simulador básicamente inicia con un vector de tiempo (t) y dos vectores de parámetros iniciales para cada proceso (Vea Figura 2.2). Estos vectores son introducidos a un *black box* que contienen dos módulos encargados de realizar las simulaciones del digestor anaerobio y de las celdas de electrólisis microbianas. El resultado de este proceso es la generación de *Acetato* y H_2 en el tiempo t_n. Este proceso se repite t veces, donde

$t = rango(0, total\_días, separación)$. Así, con 3600 días de simulación y una separación de una hora, así el proceso se repetirá 86402 veces. En otras palabras, t_1 representa la medición en la hora 1, teniendo 24 mediciones por cada día simulado.

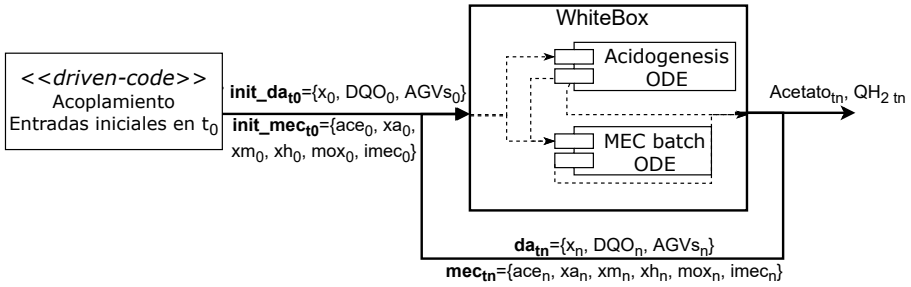

Figura 2.2: Simulador de acoplamiento DA-MEC

Ahora bien, además de las variables H_2 y *Acetato*, existen otras más que resultan interesantes monitorear. No sólo por la capacidad que tiene cada una para aportar información al funcionamiento del proceso, sino también por el hecho de ser variables críticas en la detección de fallas. Por ejemplo, la producción de *Acetato* depende de la concentración de materia orgánica (en términos de la demanda química de oxígeno o dqo_in) que llega a la entrada del DA para ser degradada. Detectar un funcionamiento anormal en esta variable es vital para tener una biorrefinería sana. Dicho esto, tomando en cuenta la opinión del experto en el tema, se han seleccionado una serie de variables en cada proceso para su monitoreo y análisis de fallas. La variables seleccionadas por cada proceso se pueden observar en la Figura 2.3.

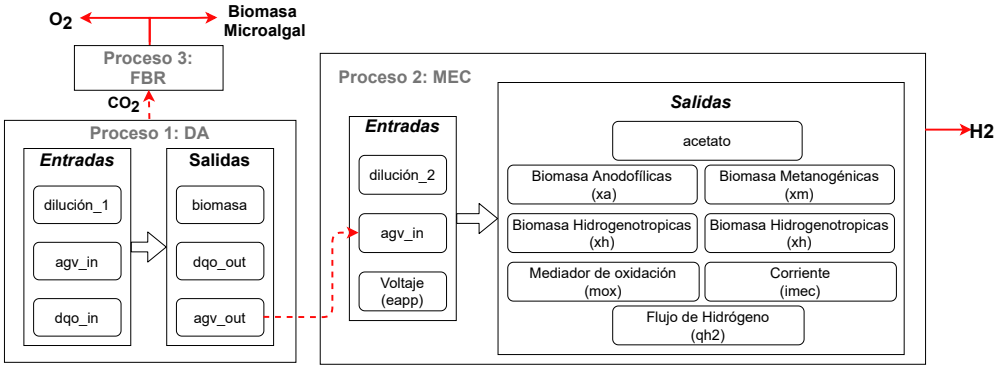

Figura 2.3: Variables dentro de un simulador para una biorrefinería con tres procesos

2.3 Implementación del simulador

El simulador descrito previamente fue originalmente implementado en el lenguaje de programación MATLAB. Sin embargo, para propósitos de este proyecto se realizó una

implementación en el lenguaje Python tomando como base la versión de MATLAB.

Python es un lenguaje de programación interpretado, multiparadigma, de alto nivel y de propósito general. Uno de los paradigmas soportados es el orientado a objetos (POO). Los objetos son entidades que tienen un determinado "estado" y "comportamiento (método)". POO nos permite tener un código más organizado, modular y robusto.

Las clases identificadas son Variable, Proceso y Biorrefinería (Figura 2.4. Una variable tiene un valor inicial, un mínimo y un máximo. Un proceso puede tener nombre, variables (entradas y salidas). Una biorrefinería esta formada de procesos. Por otra parte, un proceso tiene un comportamiento específico descrito por una serie de ecuaciones diferenciales ordinarias (ODE, por sus siglas en ingles). Las ODEs permiten describir las dinámicas de la biorrefinería.

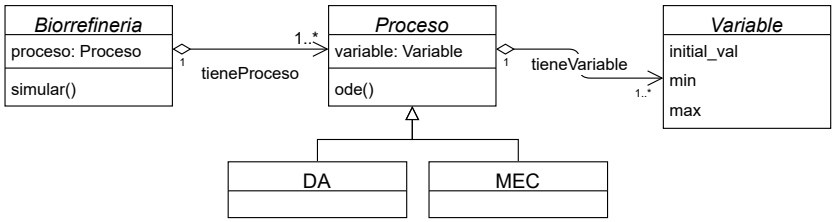

Figura 2.4: Clases para el simulador

2.3.1 Simulación

La simulación se da a través de la clase Biorrefineria la cual modela el acoplamiento de los procesos DA y MEC descrito previamente. Está clase se encarga de ejecutar la simulación a través de la integración de las ODEs de cada proceso.

El proceso se ilustra se ilustra en la Figura 2.5. De forma general, el primer paso es crear las variables propias de cada proceso, donde a cada variable se le define el valor inicial de simulación. Luego, se crean los procesos involucrados en la simulación y a cada uno se les define cuáles son sus variables de entrada y de salida. Posteriormente, se crea una instancia de la clase Biorrefineria definiendo los días de simulación. Seguido, se inicializan las variables de cada proceso con el tiempo de simulación definido. Por último, se agregan los procesos ya listos a la biorrefinería creada y se procede a la simulación. La simulación está caracterizada por un ciclo principal, el cual itera sobre las unidades de tiempo definidas por el total de días de simulación. Por cada unidad de tiempo se ejecutan las ODEs de cada proceso y se almacena los resultados.

El código completo se puede consultar en: `https://github.com/christianvadillo/simulador_bio_python`

2.3.2 Resultados de la implementación

Se hicieron 10 simulaciones por cada una de las implementaciones. Se registró el resultado de las salidas de la variables para un chequeo de sanidad y el tiempo de simulación por cada evento. Como se puede observar en la Tabla 2.1 y 2.2 las salidas de nuestra implementación

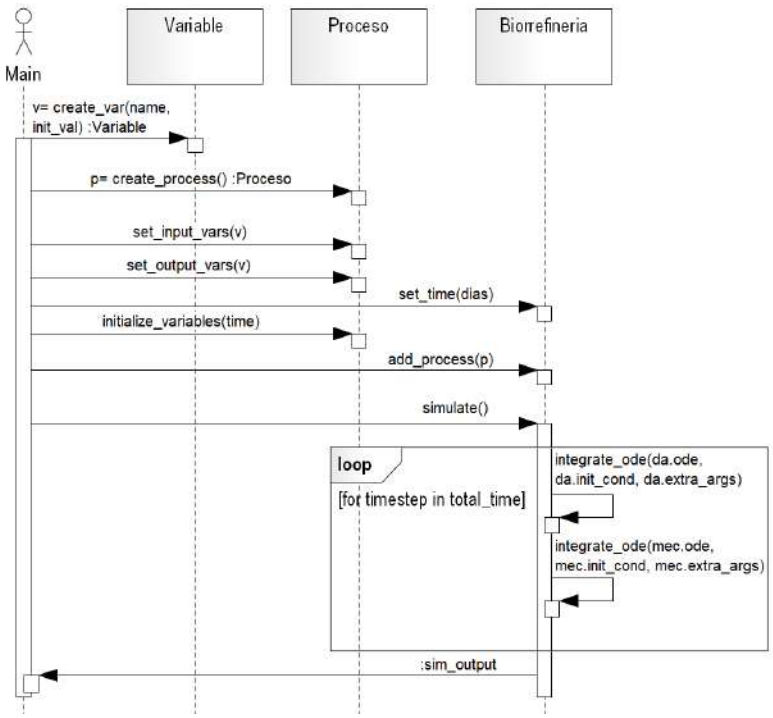

Figura 2.5: Proceso de simulación

en Python con respecto a la implementación original en MATLAB no es idéntica, debido a la diferencia en la implementación de las ODEs que cada lenguaje hace, pero las diferencias entre cada variable son diminutas. En otras palabras, el simulador en Python captura correctamente el comportamiento esperado de la simulación.

			Valor promedio de salida	
Variable	Unidad	Descripción	Python	Matlab
biomasa	gL^{-1}	Materia orgánica de origen vegetal o animal	59.956462	59.9596
dqo_out	gL^{-1}	Dióxido de carbono	14.213040	14.2128
agv_out	$mmolL^{-1}$	Concentración de acetato	140.939781	140.9430

Tabla 2.1: Valor final promedio para las variables de salida en el proceso DA para una simulación de 3600 días

Con respecto al tiempo de ejecución del proceso acoplado (DA-MEC), podemos ver en la Tabla 2.3 que la implementación en Python supera por mucho a la hecha en MATLAB, siendo aproximadamente 11.5 veces más rápido.

			Valor promedio de salida	
Variable	Unidad	Descripción	Python	Matlab
ace_out	$mmolL^{-1}$	Concentración de acetato dentro de la MEC	5690.542508	5690.7000
xa	mgL^{-1}	Biomasa anodofílica	732.038253	732.0475
xm	mgL^{-1}	Biomasa metanogénica	0.026569	0.0266
xh	mgL^{-1}	Biomasa hidrogenotrópica	0.011414	0.0114
mox	L^{-1}	Mediador de oxidación	0.101490	0.1013
imec	A	Corriente	0.046200	0.0462
qh2	Ld^{-1}	Flujo de hidrógeno producido	0.455428	0.4554

Tabla 2.2: Valor final promedio para las variables de salida en el proceso MEC para una simulación de 3600 días

	Tiempo de Ejecución (s)	
Simulación	Python	Matlab
1	28.880993	355.931666
2	27.186598	314.129656
3	27.246132	304.245377
4	26.945729	310.033032
5	27.307899	313.979148
6	28.653992	315.724433
7	26.787527	311.426489
8	26.933944	313.131862
9	26.739461	315.687116
10	28.038409	314.887149
Promedio	27.472068	316.917593

Tabla 2.3: Tiempo de ejecución por cada simulación de 3600 días

2.4 Obtención de datos anómalos

Uno de los objetivos planteados para este proyecto es el de desarrollar un sistema que detecte anomalías y emita recomendaciones. Sin embargo, como se comentó en la sección previa, no se cuenta con datos reales del funcionamiento de la biorrefinería y, si bien se tiene un simulador para el acoplamiento de dos de los tres procesos, este simulador solo genera datos de una biorrefinería sana y en correcto funcionamiento. En otras palabras, en su estado original es incapaz de representar comportamientos anormales de la biorrefinería y sus procesos. Además, carece de aleatoriedad, es decir, el resultado siempre es el mismo sin importar cuantas veces se ejecute la simulación.

Esto representa un problema dado que los métodos utilizados para el modelado de los sistemas de detección de anomalías, que se discutirán posteriormente, requieren en gran medida de los datos de la biorrefinería funcionando en un estado anormal. Por consiguiente, previo a la creación de los modelos, se realizó un proceso de generación de datos anómalos.

La idea es generar información lo más apegada a la realidad. Por lo tanto, solicitamos la supervisión de un experto en el proceso acoplado DA-MEC tema para la validación de los errores inducidos sistemáticamente.

2.4.1 Límites de variables

			Valor de salida		**Etiqueta para error**	
Variable	Unidad	Descripción	Mínimo	Máximo	Mínimo	Máximo
da_dil1	Ld^{-1}	Tasa de dilución	0.31	1	1	2
da_dqo_in	gL^{-1}	Dióxido de carbono	10	80	3	4
da_agv_in	$mmolL^{-1}$	Concentración de acetato	50	150	5	6
da_biomasa	gL^{-1}	Materia orgánica de origen vegetal o animal	0	415.176	7	8
da_dqo_out	gL^{-1}	Dióxido de carbono	2.481	44.997	9	10
da_agv_out	$mmolL^{-1}$	Concentración de acetato	3002.6	26061.4	11	12
mec_agv_in	$mmolL^{-1}$	Concentración de acetato	3002.6	26061.4	13	14
mec_dil2	Ld^{-1}	Tasa de dilución	1	3	15	16
mec_eapp	V	Voltaje aplicado	0.1	0.6	17	18
mec_ace	$mmolL^{-1}$	Concentración de acetato dentro de la MEC	0	23284.9	19	20
mec_xa	mgL^{-1}	Biomasa anodofílica	0	1396.26	21	22
mec_xm	mgL^{-1}	Biomasa metanogénica	0	0.00017	23	24
mec_xh	mgL^{-1}	Biomasa hidrogenotrópica	0	14.3591	25	26
mec_mox	L^{-1}	Mediador de oxidación	0	0.00889	27	28
mec_imec	A	Corriente	0	0.0881	29	30
mec_qh2	Ld^{-1}	Flujo de hidrógeno producido	0	0.86851	31	32

Tabla 2.4: Máximos y mínimos para variables de monitoreo en DA y MEC

El primer paso para la generación de errores fue encontrar puntos de referencia para determinar un comportamiento anormal. Para este caso se tomaron los límites máximos y mínimos para cada una de las variables de entrada elegidas para el monitoreo (véase Figura 2.3). Por cada una de las variables se realizó una simulación con los mínimos y máximos siendo tratados como constantes. El resto de las variables permanecieron igual, sin ser alteradas. Posteriormente, siguiendo la misma línea, se realizaron simulaciones con combinaciones de estas variables de entrada, manteniendo siempre sus valores constantes y el resto sin alterar. El objetivo era conseguir información del comportamiento de las otras variables no alteradas cuando estas alcanzan su estado estacionario, es decir, cuando la variación en sus valores es cero.

Este proceso nos permitió identificar los límites de operación para las variables de salida. Los resultados obtenidos se pueden ver en la Tabla 2.4

2.4.2 Generador de errores

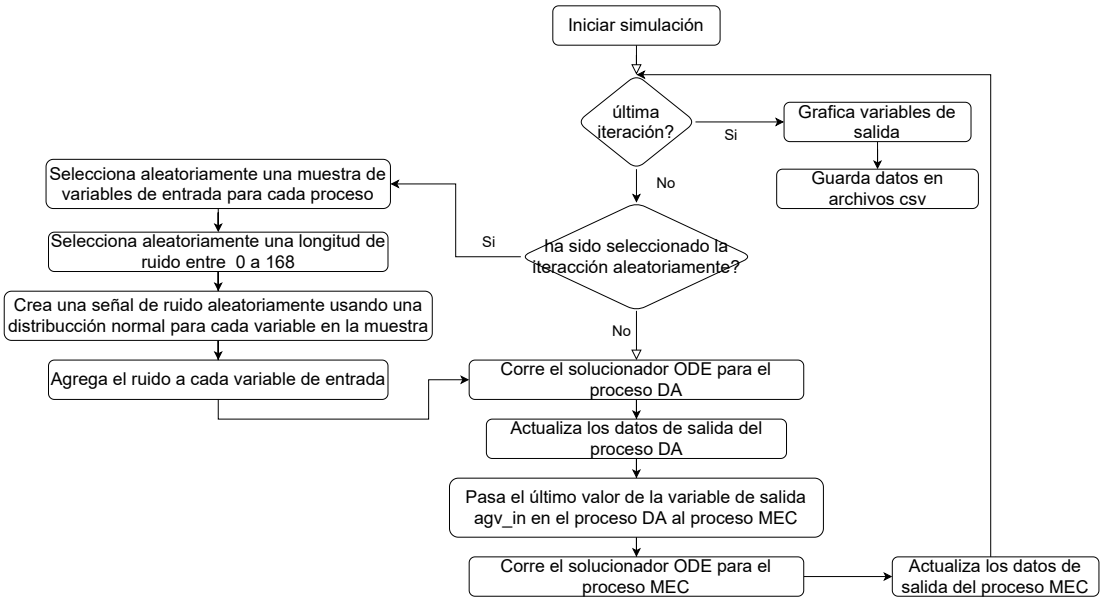

Figura 2.6: Proceso de simulación con inyección aleatoria de ruido

Teniendo como referencia los máximos y mínimos calculados previamente, el siguiente paso fue la generación de errores. Para ello, primero se debió introducir componentes de aleatoriedad a las variables de los procesos.

La inyección de aleatoriedad se hizo en dos pasos. Primero, al momento de inicializar las variables de entrada se solicita una inyección de ruido siguiendo una distribución normal. Esta distribución tiene una media igual al valor inicial definido de la variable y una desviación estándar definida por el usuario. Segundo, durante la ejecución de la simulación se inyecta de forma aleatoria ruido a las variables de entrada. En la Figura 2.6 se observa el flujo de simulación con la inyección de ruido agregada. Esta segunda inyección genera ruido mayor a dos desviaciones estándar, lo que puede llegar a inducir a que las variables sobrepasen sus límites establecidos anteriormente y producirse valores anómalos.

De esta forma se inducen estados anormales a través de las variables de entrada. Posteriormente, debido a que cada fila representa una medición registrada en una hora, se realizó un barrido sobre las filas de cada conjunto de datos resultante (Figura 2.7). Con esto determinamos si se han sobrepasado los límites máximo y mínimo de las variables. Si se determina que una variable pasó sus límites, se asigna a la medición el número del error correspondiente con base a la Tabla 2.4)

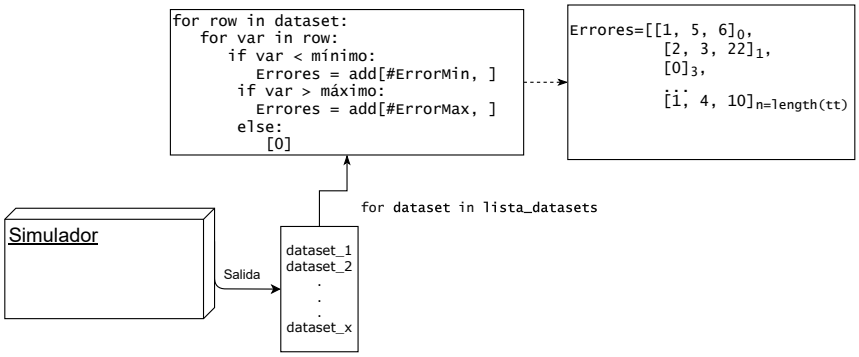

Figura 2.7: Extracción de errores para cada conjunto de datos resultante de las simulaciones con entradas constantes

Error	Descripción	count	Recomendación (Propuesta)
[]	Normal	1222589	-
[1]	Dil1 bajo	18601	Aumentar el flujo de entrada
[2]	Dil1 alto	186010	Disminuir el flujo de entrada
[3]	AGV_in_DA bajo	74404	Verificar la concentración de AGV en la entrada
[4]	AGV_in_DA alto	74404	Verificar la concentración de AGV en la entrada
[5]	DQO_in_DA bajo	18601	Verificar la carga orgánica en la entrada
[6]	DQO_in_DA alto	18601	Verificar la carga orgánica en la entrada
[15]	Dil2 bajo	18601	Aumentar el flujo de entrada
[16]	Dil2 alto	55803	Disminuir el flujo de entrada
[17]	Eapp bajo	12935	Aumentar el voltaje aplicado a la MEC
[18]	Eapp alto	74404	Disminuir el voltaje aplicado a la MEC
[1, 9]	Dil1_bajo ->DQO_out_bajo	37202	Revisar bombas e entrada para aumentar el flujo de entrada,revisar temperatura y pH dentro del reactor.
[5, 9]	DQO_in_DA_bajo ->DQO_out_bajo	14829	Revisar la concentración de DQO en la entrada, temperatura y pH
[5, 7, 9]	DQO_in_DA bajo -> Biomasa_DA_Bajo, DQO_out_DA_bajo	3772	Revisar la concentración de DQO en la entrada, temperatura y pH, revisar bombas de entrada y salida.
[6, 8]	DQO_in_DA alto ->Biomasa_DA_alto	55803	Revisar la concentración de DQO de entrada.
[13, 28]	AGV_in_MEC bajo ->mox_alto	18601	Verificar la concentración de AGV en la entrada.
[14, 20]	AGV_in_MEC_alto ->Ace_out_MEC_alto	55803	Revisar concentración de AGV en la entrada, revisar bomba de entrada.
[15, 22, 26, 28, 31]	Dil2_bajo -> xa_alto, xh_alto, mox_alto, QH2_bajo	37202	Revisar bomba de entrada para aumentar el flujo de entrada.
[15, 22, 24, 26, 31]	Dil2_bajo -> xa_alto, xm_alto, xh_alto, QH2_bajo	910	Revisar bomba de entrada para aumentar el flujo de entrada.
[15, 22, 26, 31]	Dil2_bajo -> xa_alto, xh_alto, QH2_bajo	17691	Revisar bomba de entrada para aumentar el flujo de entrada.
[15, 26, 31]	Dil2_bajo -> xh_alto, QH2_bajo	18601	Revisar bomba de entrada para aumentar el flujo de entrada.
[16, 28]	Dil2_alto ->mox_alto	30528	Revisar bomba de entrada para disminuir el flujo de entrada.
[17, 20, 21, 27, 29, 31]	Eapp_bajo-> Ace_out_MEC_alto, xa_bajo, mox_bajo, imec_bajo, QH2_bajo	24265	Revisar voltaje aplicado para aumentar el potencial aplicado y concentración de AGV de entrada.
[17, 20, 21, 27, 29]	Eapp_bajo-> Ace_out_MEC_alto, xa_bajo, mox_bajo, imec_bajo	2	Revisar conexiones eléctricas y resistencias parásitas
[18, 22, 30, 32]	Eapp_alto-> xa_alto, imec_alto, QH2_alto	37202	Revisar voltaje aplicado para disminuir el potencial aplicado y revisar el voltaje aplicado.
[18, 22, 28, 30, 32]	Eapp_alto-> xa_alto, mox_alto, imec_alto, QH2_alto	37202	Revisar voltaje aplicado para disminuir el potencial aplicado y revisar el voltaje aplicado.
[20, 21, 27, 29, 31]	Ace_out_MEC_alto, xa_bajo, mox_bajo	60875	Revisar voltaje aplicado bajo, conexiones eléctricas y resistencias parásitas.
[20, 21, 27, 29]	Ace_out_MEC_alto, xa_bajo, mox_bajo, QH2_bajo	2	Revisar voltaje aplicado bajo, conexiones eléctricas y resistencias parásitas, revisar posibles fugas en el reactor.

Tabla 2.5: Errores validados por el experto

Por último, con el fin de tener errores cercanos a la realidad, se solicitó al experto en el proceso que validará los resultados obtenidos. De esta forma, el conjunto de errores utilizados para la creación de los modelos que se introducirán en capítulos posteriores se puede ver en la Tabla 2.5.

2.5 Estructura del sistema de monitoreo y detección de anomalías

La arquitectura general de la plataforma (Figura 2.8) está compuesta por tres módulos principales: el sistema de monitoreo, el de detección de anomalías y el sistema de recomendaciones. La plataforma estará trabajando en un servidor web que recibirá información o mediciones de la biorrefinería. Por otra parte, debido a que aun no se cuenta con información de los sensores en línea, ni de la red de sensores en línea en los procesos de la biorrefinería, supondremos que la información será registrada por un técnico capacitado en una base de datos cada cierto tiempo. Así, la plataforma está pensada para cumplir tres objetivos principales:

- Monitorear las mediciones registradas.
- Detectar una anomalía o falla.
- Emitir una recomendación.

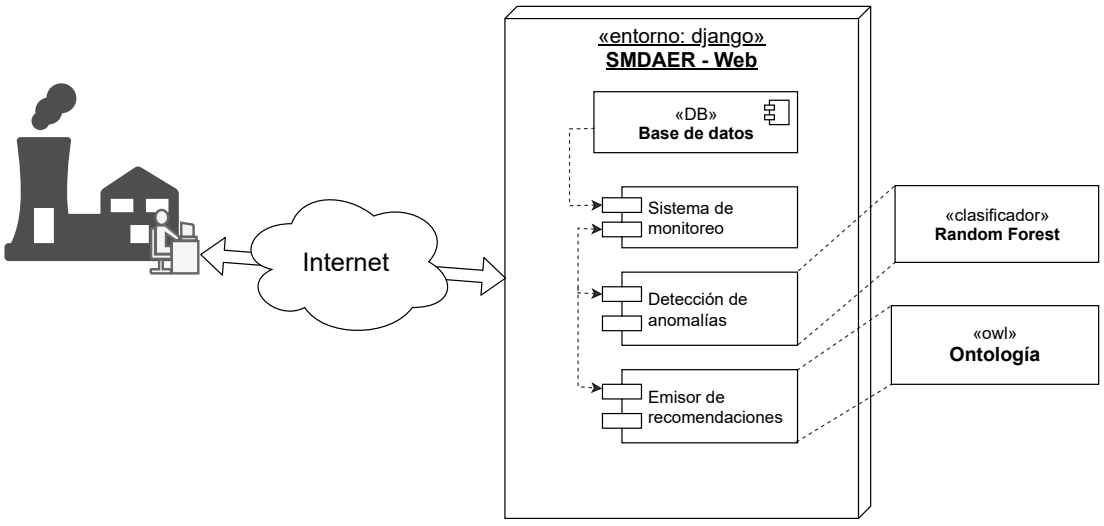

Figura 2.8: Arquitectura general de la plataforma para el monitoreo, detección de anomalías y emisión de recomendaciones

2.5.1 Random Forest

Random Forest es un método de aprendizaje automático que es entrenado con un conjunto de datos que contienen tanto mediciones de un funcionamiento normal, como mediciones anormales. La idea es crear un modelo utilizando los datos (errores y normales) generados por el simulador. Este modelo realiza la labor de detectar anomalías en las mediciones de los procesos. En el Capítulo 3 se profundizará más sobre el desarrollo del modelo y su rendimiento.

2.5.2 Ontología

Posterior al primer diagnóstico hecho por el modelo RF, se aplicará un modelo basado en representación del conocimiento para obtener un segundo diagnóstico. Este se basa en un modelo cualitativo que representa el conocimiento a priori de los procesos bajo monitoreo. El diagnóstico de fallas se realiza ejecutando una serie de reglas semánticas bien definidas. De encontrarse alguna anomalía o falla, el modelo es capaz de extraer información sobre la variable, el proceso y la falla que se está presentando. El modelo se expicará con profundidad en el Capítulo 4

2.5.3 Aplicación web

Una de las ventajas de una aplicación web es que su implementación suele consistir en desarrollar y configurar los componentes en el lado del servidor. Es decir, no requiere ningún software o configuración especial por parte del cliente.

La aplicación web de la plataforma será el contenedor de todos los micro-sistemas que, en conjunto, darán la funcionalidad de monitoreo y detección de anomalías. Para la implementación se ha elegido *Django* como *framework* de diseño. Django facilita a los desarrolladores la producción de aplicaciones que sean fáciles de mantener y funcionen correctamente bajo carga [43]. El desarrollo de la aplicación web será explorado en el Capítulo 5

Capítulo 3

Detección de anomalías en sistemas complejos

En la actualidad, los seres humanos dependen de la operación de diversos sistemas. Por ejemplo, los sistemas de transporte (trenes, barcos y aviones), los sistemas de comunicación (televisión, teléfono y redes informáticas), sistemas de servicios públicos (agua, gas y electricidad), sistemas industriales (plantas de fabricación, químicas y de tratamientos residuales), por mencionar algunos. Por sus características internas y funcionales, a estos tipos de sistemas se les cataloga como sistemas complejos [44].

Este capítulo tiene dos secciones principales. La primera sección presenta el marco teórico del uso de modelos para detección de anomalías en sistemas complejos; además, se presenta el estado de arte sobre técnicas usadas para sistemas de detección de anomalías. La segunda sección del capítulo presenta el desarrollo de un modelo de clasificación de anomalías utilizando un método de aprendizaje *Random Forest*. El objetivo principal es crear un sistema que sea capaz de diferenciar un estado de funcionamiento normal de uno anormal utilizando datos provenientes de las mediciones hechas en los procesos.

3.1 Marco teórico

3.1.1 Sistema complejo

En la literatura es posible encontrar varias definiciones de un sistema complejo [45–48]. Pero a grandes rasgos, es aquel que está formado por una gran cantidad de elementos [46]. Estos elementos, pueden ser aves, insectos, personas, líneas de código o células, y su cantidad puede ser de miles o millones. Además, intercambian información y constantemente están interactuando entre si de forma no lineal [49]. A menudo tienen niveles de organización que pueden considerarse como una jerarquía de sistemas y subsistemas, como lo propuso Herbert Simon en su artículo *The Architecture of Complexity* [50]. Sin embargo, la propiedad más llamativa de los sistemas complejos es la existencia de fenómenos emergentes [45]. Son fenómenos nuevos que se presentan cuando un sistema adquiere cierto nivel de complejidad,

y que originan nuevos niveles de organización [44]. Estos fenómenos emergentes no pueden simplemente derivarse o predecirse únicamente a partir del conocimiento de la estructura de los sistemas y las interacciones entre sus elementos individuales [51].

Las características señaladas anteriormente plantean una serie de desafíos para entender y controlar un sistema complejo. Uno de los problemas más críticos que rodean a los sistemas complejos es la seguridad y confiabilidad del sistema. La teoría del control, una disciplina que se ocupa de los fundamentos formales del análisis y diseño de los sistemas de control y gestión de sistemas [12], establece que todo sistema construido por humanos es poco confiable [13]. La degradación con el tiempo y/o el uso producen fallas en los procesos del sistema. Algunas fallas pueden tener un impacto negativo. Por ejemplo, pérdidas humanas por fallas en sistema del sector salud [52] o perdidas millonarias en accidentes aeroespaciales [53]. Siendo más específicos, en sistemas complejos como las plantas industriales, una falla puede causar pérdidas millonarias de dinero y graves daños al medio ambiente [14–17]. Una forma común de afrontar estos problemas es mejorar la calidad, confiabilidad y robustez de los componentes individuales, como sensores, actuadores, controladores o computadoras. Sin embargo, aún mejorando cada componente individual, no es posible garantizar un sistema totalmente libre de fallas funcionales [18]. Así, a medida que los sistemas industriales modernos crecen y se vuelven complejos, el uso oportuno de sistemas de monitoreo y diagnóstico de fallas se hace más relevante [19]. Estos nos permiten garantizar la seguridad de los procesos en los sistemas [2], al monitorear los niveles óptimos definidos; facilitar la toma de decisiones [20], al tener una mejor comprensión del comportamiento del sistema industrial; y mejorar la calidad del producto industrial [21], al tener funcionando en óptimas condiciones los procesos del sistema. Así, dada la incapacidad de tener un sistema totalmente seguro y confiable, los sistemas de monitoreo de procesos y detección de anomalía han captado la atención tanto del sector académico e industrial en los últimos años [22–30], incluida la detección, identificación y el diagnóstico de fallas.

3.1.2 Detección de anomalías

Muchos procesos en las industrias químicas experimentan condiciones anormales que conducen a productos que no cumplen con especificaciones establecidas o al bloqueo del proceso [54]. A menudo, un comportamiento anómalo suele ser más interesante y contener información más importante que un comportamiento normal.

Una anomalía, o valor atípico, se define como una observación que es inconsistente con el resto de un conjunto de datos [31]. Por otra parte, la detección de anomalías se puede definir como un proceso cuyo objetivo es encontrar patrones o datos que no encajen con el comportamiento esperado dentro de un conjunto general [5]. Si se identifica un comportamiento anormal, el siguiente paso es encontrar la causa de la anomalía o el diagnóstico de falla. El diagnóstico consiste en determinar qué anomalía ocurrió, en otras palabras, determinar la raíz de la causa del estado fuera de control observado [54]. El diagnóstico se puede llevar a cabo asociando patrones de comportamiento del proceso a fallas específicas ya definidas. Por ejemplo, relacionar las variables del proceso que tienen desviaciones significativas de sus valores esperados con varios estados del proceso. Por

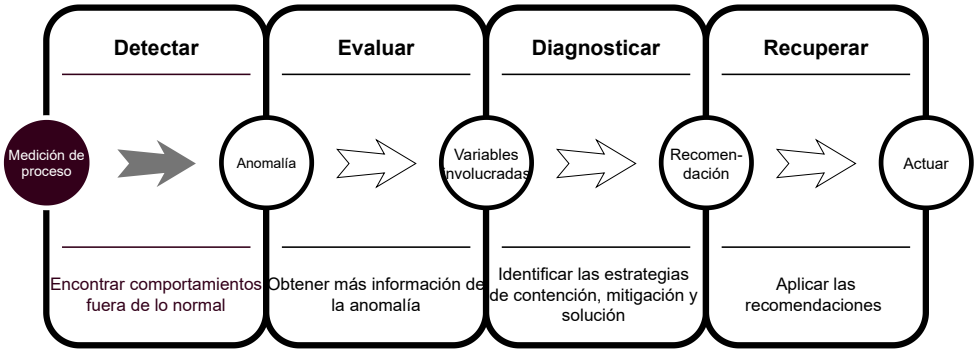

Figura 3.1: Flujo completo detección de anomalías

último, ya con la falla diagnosticada, se procede a la recuperación del proceso, también llamada intervención, que consiste en eliminar el efecto de la anomalía. El propósito de este procedimiento es enfocar la atención del operador de la planta y del ingeniero en los subsistemas más pertinentes, de modo que el efecto de la anomalía pueda eliminarse de una manera más eficiente [55].

Las características que un método de detección de anomalías requiere tener son las siguientes [56]. Primero, debe poder identificar con precisión todos los tipos de anomalías, así como el comportamiento normal del proceso. Segundo, debe ser robusto, es decir, debe poder manejar los cambios de patrones en los conjuntos de datos. Tercero, es deseable que el método de detección pueda descubrir anomalías en tiempo real o casi en tiempo real. Por último, debe también poder detectar múltiples fallas de forma simultánea.

Así, la idea básica en detección de anomalías es ver y monitorear tantos procesos críticos como sean posibles en busca de patrones anómalos. Para esto, es esencial desarrollar procedimientos sistemáticos y computacionalmente eficientes para descomponer un sistema complejo en modelos [57]. El interés radica en obtener modelos que puedan ser implementables en dispositivos informáticos. Esto permite conducir una aplicación práctica de los modelos, por ejemplo, en sistemas de monitoreo y detección de fallas.

Una clasificación básica de métodos para modelar sistemas de monitoreo y detección de fallas sería: modelado clásico y modelado basado en técnicas de *data-driven* [58].

3.1.3 Modelado clásico

El modelado clásico típicamente considera los fenómenos naturales o físicos del sistema. Se derivan expresiones algebraicas paramétricas que cuantifican una relación funcional entre variables de interés. En particular, emplean ecuaciones diferenciales e integrales cuyas soluciones son señales analógicas [59]. Con el modelado clásico es posible obtener información importante sobre la interacción entre los subsistemas y sus interrelaciones. También podemos llegar a interpretaciones físicas importantes de las acciones de los elementos en términos de resultados globales [57].

Los enfoques basados en modelos clásicos han recibido una atención considerable, en particular cuando se dispone del conocimiento físico o matemático de los sistemas.

Por ejemplo, en los procesos químicos [60], las redes de sensores [61–63] o en el sector transporte [64]. Trabajos de investigación dedicados a revisar y analizar el estado del arte de las técnicas basados en modelos se pueden encontrar en [18, 65, 66]

Sin embargo, la modelación clásica, en general, es compleja, difícil de realizar e implementar [57]. Requiere tener una comprensión profunda del sistema que se está modelando. Además, el modelado clásico está restringido a problemas similares [59]. De hecho, hay modelos prácticos donde es difícil [18, 64, 67] (o incluso imposible) identificar los fenómenos naturales de los sistemas. En tales casos, una modelación clásica claramente no es una estrategia adecuada [46].

3.1.4 Modelado usando técnicas de *data-driven*

En años recientes, el modelado de sistemas utilizando técnicas de *data-driven* ha sido empleado [19, 68–70] como una alternativa legítima cuando no se pueden aplicar modelados clásicos. La diferencia con la modelación clásica es que no requiere conocer los principios físicos que rigen al sistema.

En general, se utiliza la información recolectada a través del tiempo de un proceso durante condiciones de operación normal [23]. Posteriormente, se desarrollan medidas para detectar e identificar condiciones anormales. Los datos recolectados durante fallas específicas se usan para aprender medidas que permitan el diagnóstico de anomalías [22]. Debido a que estos métodos se basan en los datos, dependen en gran medida de la cantidad y la calidad de los mismos [33]. Sin embargo, dado al creciente uso de sensores en los sistemas, tales como, sensores de temperatura, humedad, presión, nivel y flujo, cada vez es más sencillo contar con grandes volúmenes de datos, lo que facilita la aplicación de estas técnicas.

Entre los esquemas más utilizados están las técnicas de análisis multivariable (MVA). Debido a su simplicidad y eficiencia en el procesamiento de grandes cantidades de datos, son reconocidas como herramientas robustas que utilizan métodos estadísticos para abordar el monitoreo de procesos y diagnóstico de problemas [71]. En esta categoría se destacan *Principal Component Analysis* [19, 22, 28, 69, 72, 73], *Partial Least Squares* [28, 34, 74–76] y *Fisher's Linear Discriminant Analysis* [34, 74, 75, 77].

En las técnicas multivariadas el modelo crece solo con la complejidad del mismo, no con el tamaño de los datos [78]. Por lo tanto, permiten que el modelo se evalúe rápidamente en nuevas instancias, lo que los hace adecuados para grandes conjuntos de datos. Si se sabe que los datos se ajustan a un modelo de distribución estadístico y que esta distribución no cambiará durante el tiempo de vida del sistema, estos enfoques son una buena opción para el monitoreo y detección de anomalías [79]. Sin embargo, debido a que su aplicabilidad depende del ajuste a un modelo estadístico, que a menudo no es constante en los sistemas complejos, se dificulta su aplicabilidad [31].

En años más recientes han surgido métodos que no requieren el tener una distribución estadística asociada al sistema. Estos incluyen técnicas de aprendizaje automático e inteligencia artificial, tales como *Independent component analysis* [80–82], *Artificial Neural Networks* (ANN) [27, 83, 84], *Support Vector Machines* (SVM) [54, 85–88].*Rule-Based* [3, 4, 89, 90],

Rule-Based Classifiers [3, 4, 89, 90] y *Random Forest* (RF) [6, 7, 91]. Estos métodos se encuentran dentro de la clase de métodos de regresión/clasificación. Aunque los métodos de aprendizaje automático llegan a ser bastante robustos en general, incluso con datos de procesos, no proporcionan modelos únicos, ni alguna forma de causalidad, es decir, no pueden identificar el efecto de una variable en el sistema completo [33]. En otras palabras, se dificulta la evaluación y diagnóstico de una posible falla. Además, sus implementaciones suelen tener un alto costo computacional y son poco eficientes cuando se desea detectar múltiples anomalías de forma simultánea [33].

3.2 *Random Forest*

Random Forest es un método de aprendizaje supervisado, es decir, requiere de tener un conjunto de datos ya etiquetados para su entrenamiento. Este método construye múltiples árboles de decisión, donde cada árbol funciona en una muestra aleatoria del conjunto de datos. Por lo tanto, para comprender mejor el método de *Random Forest*, iniciaremos hablando de los árboles de decisión.

3.2.1 Árboles de decisión

Un árbol, computacionalmente hablando, es un grafo $G = (V, E)$ en el que cualquier par de nodos o vértices (V) están conectados por exactamente un camino o arista (E) [92]. Como estructura de datos, un árbol está hecho de una colección de nodos y aristas organizados de forma jerárquica (Figura 3.2). Los nodos se dividen en nodos internos (o divididos) y nodos terminales (u hojas). Todos los nodos (excepto la raíz) tienen exactamente un borde entrante. Además, a diferencia de los grafos generales, un árbol no puede contener bucles [93]. Para esclarecer mejor la idea de los árboles de decisión, definamos los siguientes conceptos [92]:

Definition 3.2.1. Un **árbol con raíz** *es un árbol en el cual uno de sus nodos ha sido designado como* ***raíz****. Además, un árbol con raíz es un grafo dirigido donde todas las aristas son dirigidas lejos de la raíz.*

Definition 3.2.2. *si existe una arista que va de t_1 a t_2 (por ejemplo, si $(t_1, t_2) \in E$), entonces se dice que el nodo t_1 es* ***padre*** *del nodo t_2, mientras que el nodo t_2 se dice que es* ***hijo*** *del nodo t_1*

Definition 3.2.3. *en un árbol con raíz, un nodo se dice que es* ***interno*** *si tiene uno o más hijos, y es* ***terminal*** *si no tiene hijos. Los nodos terminales también son llamados* ***hojas***

Definition 3.2.4. *Un* ***árbol binario*** *es un árbol con raíz donde todos sus nodos internos tiene exactamente dos hijos*

En estos términos, un árbol de decisión puede ser definido como un modelo representado por un árbol con raíz (generalmente binario, pero no necesario), donde cualquier nodo t

representa un subespacio $X_t \subseteq X$ del espacio de entrada, con el nodo raíz t_0 correspondiendo a la propia X. Los nodos internos t son etiquetados con una división s_t tomados de un conjunto de preguntas o pruebas $\mathcal{Q}$. Entonces, el espacio X_t es dividido tal que, el nodo t se representa en subespacios separados respectivamente que corresponden a cada uno de sus hijos. Por ejemplo, el conjunto de todas las divisiones binarias en el conjunto $\mathcal{Q}$ de preguntas s de la forma ¿es $x \in X_A$?, donde $X_A \subset X$ es un subconjunto del espacio de entrada. Cualquier división s de esta forma, divide X_t en dos subespacios respectivamente. De tal forma que el hijo izquierdo corresponde a $X_t \cap X_A$ y al derecho le corresponde $X_t \cap (X \setminus X_A)$ [92]

Básicamente, se tiene un conjunto de pruebas de diagnóstico organizadas jerárquicamente en una estructura de árbol. Para un objeto de entrada dado, un árbol de decisión estima una propiedad desconocida del objeto haciendo preguntas sucesivas sobre sus propiedades conocidas. La pregunta que se debe hacer a continuación depende de la respuesta de la pregunta anterior y esta relación se representa gráficamente como una ruta a través del árbol que sigue el objeto. La decisión se toma en función del nodo terminal alcanzado por el objeto de entrada [93]. Por lo tanto, la clave para un buen funcionamiento de un árbol de decisión es establecer las pruebas asociadas con cada nodo interno y los predictores de toma de decisiones asociados con cada hoja. Por ejemplo, mire la Figura 3.3.

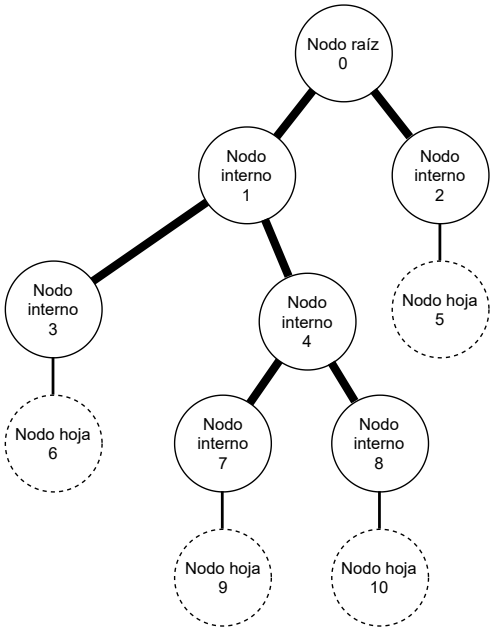

Figura 3.2: Árbol como estructura de datos

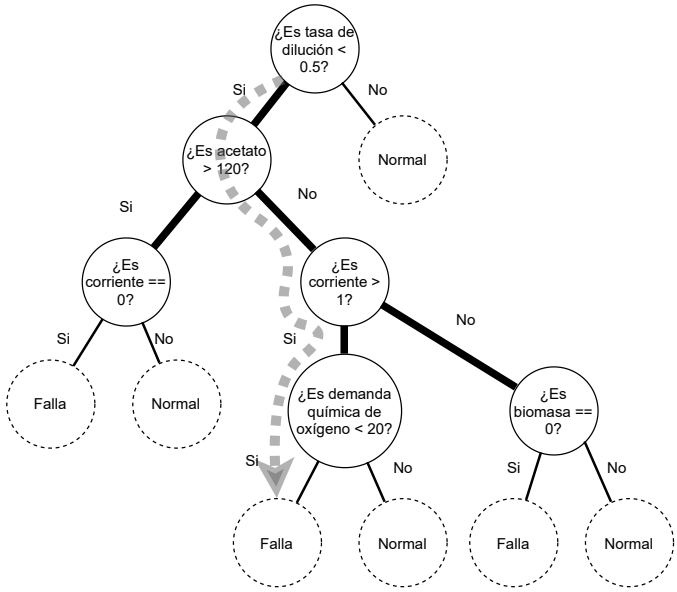

Figura 3.3: Árbol de decisión

Similar a la regresión, donde se obtiene una ecuación (por ejemplo, para predecir el precio de una casa dado sus mediciones y número de cuartos), un árbol de decisión también tiene la capacidad de predecir. Además, a diferencia de la regresión lineal o logística, que

están optimizadas para realizar un solo tipo de tarea, regresión o clasificación, los árboles de decisión pueden realizar ambos [94].

Otra forma de ver a un árbol de decisión es como una representación del conocimiento [95]. En general, los modelos expresados en un idioma se pueden traducir a otro distinto, y lo mismo ocurre con un árbol de decisión. Una traducción simple y útil es en un conjunto de reglas. Una representación en reglas tiene sus ventajas. Al revisar el conocimiento que se ha capturado, podemos considerar cada regla por separado en lugar de distraernos con la estructura más compleja de un gran árbol de decisión. Por otro lado, la estructura del árbol de decisión, es el lenguaje que usamos para expresar nuestro conocimiento. Una oración (o modelo) en este lenguaje es un árbol de decisión particular. Para cualquier conjunto de datos, habrá muchos, o incluso infinitos, posibles árboles de decisión (oraciones) [95]. Enumerarlas todas y probarlas representa un gran costo computacional [94]. Esto bien podría implicar días, semanas, meses o años dependiendo de la complejidad del problema. Así, el objetivo es utilizar las observaciones (el conjunto de datos de entrenamiento) para reducir esta tarea de búsqueda y poder encontrar un buen modelo en un tiempo razonable.

3.2.2 Particionando el conjunto de datos

El algoritmo básico para el desarrollo de un árbol de decisión se conoce como inducción de arriba hacia abajo. Utiliza un enfoque de divide y conquista, o partición recursiva [95]. Intuitivamente queremos encontrar cualquier variable de entrada que pueda usarse para dividir el conjunto de datos de entrenamiento ($\mathcal{L}$) en dos conjuntos de datos más pequeños. El objetivo es aumentar la homogeneidad de cada uno de los dos conjuntos de datos con respecto a la variable objetivo [93]. Por ejemplo, si quisieramos construir una partición del conjunto de datos original usando una variable *Dilución* con un valor dividido de 0.5, cada observación que tenga un valor menor que 0.5 se agrupan en un subconjunto. Los restantes (con *Dilución* mayor o igual a 0.5) en un segundo subconjunto. Estos nuevos conjuntos de datos tendrán 55 y 345 observaciones, respectivamente. Al dividir en esta variable, hemos hecho una mejora en la homogeneidad con las proporciones de los valores de la variable objetivo. El proceso ahora se repite nuevamente por separado para los dos nuevos conjuntos de datos y el proceso continúa repitiéndose hasta que decidamos que debemos detenernos. En general, podríamos detenernos cuando nos quedemos sin variables, sin datos, o cuando la partición del conjunto de datos no mejora las proporciones o el resultado.

Elegir el valor 0.5 para la variable *Dilución* es solo una posibilidad entre muchas opciones. Si hubiéramos elegido el valor 0.8, tendríamos dos nuevos conjuntos de datos con nuevas proporciones para las clases Normal y Falla. Esto nos hace cuestionarnos acerca de cómo elegir la mejor división de nuestro conjunto de datos de entrenamiento.

3.2.3 Criterio de medición para particiones

Una forma de elegir la mejor división del conjunto de datos es utilizar una medida qué cuantifique qué tan buena es una partición particular. Entre las más utilizadas esta la *Entropía y Ganancia de Información* [95].

En la teoría de la información, la ganancia de información asociada con un nodo dividido en un árbol se define como la reducción en la incertidumbre lograda al dividir los datos de entrenamiento que llegan al nodo en múltiples subconjuntos secundarios. Se define comúnmente de la siguiente manera [94]:

$$H(D) = -q\, log_2\,(q) - r\, log_2\,(r) \tag{3.1}$$

Aquí, $H(D)$ representa la entropía del conjunto de datos D. q es la probabilidad de que el evento 1 pase y r la probabilidad de que el evento dos suceda.

Así, podemos visualizar a un nodo raíz como el lugar donde existe la máxima incertidumbre. A medida que el árbol se divida en ramas, la incertidumbre disminuye. Por lo tanto, la elección de la división (la variable en la que se debe basar la división) depende de qué variables disminuyan más la incertidumbre o entropía [93]. Dicho de otra forma, la división que proporciona la mayor ganancia en información. Sobre la base de estos conceptos definidos, un procedimiento general para la inducción de árboles de decisión se puede describir formalmente en el Algoritmo 1 planteado en [92]

Algorithm 1 Árbol de decisión binario

procedure ConstruirArbolDeDecision($\mathcal{L}$)
  ▷ Crea un árbol de decisión φ con un nodo raíz t_0
  ▷ Crea una pila S de nodos abiertos $(t, \mathcal{L}_t)$
  ▷ $S.PUSH((t_0, \mathcal{L}))$
  while S no esté vacío **do**
    $t, \mathcal{L} = S.POP()$
    if criterio de paro se cumple **then**
      $\widehat{y}_t$ = algún valor constante
    else
      ▷ Encuentra la división en $\mathcal{L}$ que maximiza la Ganancia de Información (GI)
      ▷ Particiona $\mathcal{L}_t$ en $\mathcal{L}_{t_L} \cup \mathcal{L}_{t_R}$ de acuerdo a GI
      ▷ Crea el nodo izquierdo t_L para t
      ▷ Crea el nodo derecho t_R para t
      ▷ $S.PUSH((t_R, \mathcal{L}_{t_R}))$
      ▷ $S.PUSH((t_L, \mathcal{L}_{t_L}))$
    end if
  end while
  return φ
end procedure

3.2.4 Bosque de árboles de decisión

Como se mencionó al principio de la Sección 3.2, un conjunto de árboles de decisión constituyen un bosque de decisión aleatorio (*Random Decision Forest* o RF). RF fue descrito por primera vez por Leo Breiman [96], y fue pensado para ser un competidor directo de *Boosting* [97]. Dado la construcción en conjunto, que reduce la inestabilidad que podemos observar cuando construimos árboles de decisión únicos [95], el algoritmo

de RF tiende a producir modelos bastante precisos. Esto a menudo se puede ilustrar simplemente eliminando un número muy pequeño de observaciones del conjunto de datos de entrenamiento, para ver un cambio considerable en un árbol de decisión. Esta ventaja permite que RF sea bastante robusto al ruido (es decir, variables que tienen poca relación con la variable objetivo). Ser robusto al ruido significa que pequeños cambios en el conjunto de datos de entrenamiento tendrán poco o ningún impacto en las decisiones finales tomadas por el modelo resultante [95].

Al construir cada árbol de decisión a su profundidad máxima, como lo hace RF, aumenta la complejidad del modelo final. Esto permite obtener un modelo menos sesgado, es decir, modelos con predicciones bastantes cercanas al valor real [94]. Por otra parte, la aleatoriedad en la selección de muestras y variables predictoras o características, proporciona otro nivel de robustez ante el ruido, los valores atípicos y el sobre entrenamiento [95]. La aleatoriedad también ofrece eficiencias computacionales. Por ejemplo, durante el proceso de construcción de los árboles de decisión, en cada nodo solo se considera una pequeña fracción de todas las variables disponibles al determinar la mejor partición del conjunto de datos. Esto reduce sustancialmente el requisito computacional [95].

En resumen, RF es atractivo porque [94]:

- son relativamente rápidos de entrenar y al momento de predecir;
- tiene un excelente rendimiento en precisión en datos estructurados;
- es robusto ante el ruido y no se sobre entrena en la mayoría de los casos;
- dependen solo de uno o dos hiper-parámetros;
- puede ejecutarse en conjuntos de datos a gran escala con altas dimensiones, es decir, conjuntos de datos con una gran cantidad de variables predictoras.

3.2.5 Algoritmo de *Random Forest*

Básicamente, el algoritmo inicia construyendo árboles de decisión con muestras diferentes ($\mathcal{L}_j$) del conjunto de datos original ($\mathcal{L}$). Para cada nodo de árbol creado solo selecciona un pequeño subconjunto de variables independientes o características. A esto se le conoce como *Bagging* o *bootstrap aggregating* [94]. La aplicación de *bagging* permite que sea menos probable que las predicciones sean sesgadas debido a algunos casos atípicos.

Algorithm 2 Random Forests. Sea $\mathcal{L} = (x_1, y_1), \ldots, (x_N, y_N)$ un conjunto de datos de entrenamiento con $x_i = (x_{i,1}, \ldots, x_{i,p})^T$ y J el número de árboles en el bosque

```
1: procedure ConstruirRandomForest(L)
2:     for j=1 to J: do
3:         ▷ Tome una muestra aleatoria de arranque L_j de tamaño n de L
4:         ▷ Construya un árbol de decisión binario φ (Algoritmo 1) usando la muestra de arranque L_j
5:     end for
6: end procedure
```

Posterior a la construcción del bosque, cada vez que una nueva muestra necesite ser clasificada, esta es proporcionada a cada árbol en el bosque y cada árbol emite un voto de unidad hacia cierta clase que indica la decisión del árbol. El bosque elige la clase con más votos para clasificar a la nueva muestra [93]. Esto puede realizarse mediante una simple operación de promedio. Por ejemplo, dado un bosque con T árboles (donde $t \in 1, \ldots, T$ es el índice de cada árbol) y $\mathbf{v}$ una muestra de prueba, tenemos entonces [92]:

$$p(c \mid \mathbf{v}) = \frac{1}{T} \sum_{t=1}^{T} p_t \left(c \mid \mathbf{v}\right) \tag{3.2}$$

donde $p_t \left(c \mid \mathbf{v}\right)$ denota la distribución posterior obtenida por el t-isimo árbol.

3.3 Diseño y desarrollo del clasificador *Random Forest* para detección de anomalías

3.3.1 Trabajos relacionados

Previamente, en la Sección 3.1.4 mencionamos los algoritmos más utilizados para modelados de sistemas de monitoreo y detección de anomalías. Si bien *Principal Component Analysis* y *Support Vector Machines* han sido bastantes populares en esta área, *Random Forest* (RF) ha comenzado a ganar popularidad en años recientes [6, 7, 25, 91, 98].

En [25] se describe una técnica de visualización multidimensional para la detección de fallas y el diagnóstico de procesos multivariados en datos industriales. Se utiliza PCA para el sistema de monitoreao de eventos y *Random Forest* para el diagnóstico de fallas. RF se entrena con los datos obtenidos de todas las fallas y luego se usa para clasificar y diagnosticar fallas en tiempo real. Los resultados muestran que las tasas de detección de fallas utilizando el método propuesto mejoran considerablemente en comparación con los enfoques clásicos. Sin embargo, se requiere de más estudios y análisis en los puntajes de los componentes principales (PC) para aislar mejor las condiciones de funcionamiento anormal.

Por otra parte, en [6], los autores se enfocan en cómo usar RF para mejorar la tasa de detección de anomalías para conjuntos de datos de transmisión. En este trabajo de investigación, se profundiza en la elección correcta de los parámetros propios del modelo de RF. Primero se da una justificación matemática de la altura requerida para el árbol de decisión y el número de árboles. Luego, se diseñó una estrategia para el puntaje de votación para combinar los resultados de diferentes árboles de detección de anomalías. Finalmente, realizan el agrupamiento de características o variables predictoras correlacionadas para encontrar las anomalías determinadas conjuntamente por subconjuntos de características. Sus resultados, tanto en conjuntos de datos sintéticos, como datos reales muestran mejoras de rendimiento en la detección de anomalías.

En [7] proponen un enfoque que se basa en un nuevo tipo de algoritmo RF, adecuados para extraer reglas que expliquen la anomalía. El argumento se basa en el hecho de que solo hay unos pocos métodos de detección de anomalías que intentan explicar cómo la muestra anómala difiere del resto. La solución propuesta utiliza un tipo específico de RF

para extraer reglas que expliquen la diferencia. Luego, estas reglas se filtran y se presentan al usuario como un conjunto de reglas de clasificación que comparten el mismo consecuente, o como la regla equivalente con un antecedente en una forma disyuntiva normal. El enfoque propuesto se ha comparado ampliamente con técnicas anteriores relevante en 34 conjuntos de datos del repositorio de aprendizaje automático UCI mientras se explican las anomalías identificadas por algoritmos de detección de anomalías múltiples. Los resultados muestran que el enfoque proporciona explicaciones más ajustadas y precisas. Además, es rápido y liviano, por lo tanto, adecuado para el procesamiento de datos casi en tiempo real.

En [91] se presenta un novedoso sistema de detección de intrusos llamado TR-IDS, que aprovecha las características estadísticas y las características de carga útil. Utilizan una red neuronal convolucional de texto (Text-CNN, por sus siglas en ingles) para extraer información efectiva de las cargas útiles. Después, usando el algoritmo de *Random Forest* se realiza la combinación de características estadísticas y características de carga útil para la clasificación final. Extensas evaluaciones experimentales demostraron la efectividad del enfoque propuesto para detección de anomalías en detección de intrusos.

En [98] se analizan datos de operación industrial con algoritmos de detección de anomalías basados en series de tiempo y aprendizaje automático para descubrir ataques hacia los procesos industriales. Se utilizan dos conjuntos de datos diferentes, uno proveniente del tráfico de control de tubería de gas basado en Modbus y otro del tráfico de procesamiento por lotes basado en *Object Linking and Embedding for Process Control Unified Architecture (OPC UA)*. Para detectar ataques, se utilizan dos algoritmos basados en aprendizaje automático, a saber, *SVM* y *Random Forest*. Ambos otorgando un buen desempeño, pero con *Random Forest* ligeramente superior a *SVM*.

Si bien, todos los trabajos citados brindan enfoques eficientes utilizando RF o combinación con otras técnicas, la detección de una anomalía es solo la mitad del problema. Como humanos, tenemos más confianza a aquellos resultados en los que podamos comprender de dónde provienen. Sin embargo, de los trabajos citados solo en [7] se explora la idea de interpretabilidad del modelo. Nuestro enfoque da un paso más al utilizar modelos ontológicos para la emisión de recomendaciones (Capítulo 4) y obtención de explicaciones más cercanas al lenguaje natural.

3.3.2 Metodología empleada

Se ha seguido una metodología básica comúnmente usada para el entrenamiento de modelos de *data-driven* (véase Figura 3.4). Para iniciar, se ha obtenido un conjunto de datos provenientes del simulador introducido en el Capítulo 2. A este gran conjunto de datos lo separamos por procesos, obteniendo dos conjuntos nuevos. Así, por cada uno de estos conjuntos aplicamos un pre-procesamiento de datos y posteriormente lo dividimos en un conjunto de entrenamiento y otro de validación. Con el conjunto de entrenamiento hemos aplicado la técnica *Grid-Search* para encontrar los mejores hiperparámetros, tales como, profundidad máxima del árbol o número total de árboles en el bosque. Con los mejores hiperparámetros se entrenó el modelo RF para posteriormente realizar la validación del modelo. Finalmente, los modelos resultantes serán utilizados en un contexto “en línea”

dentro de la plataforma web para detectar estados de procesos anormales.

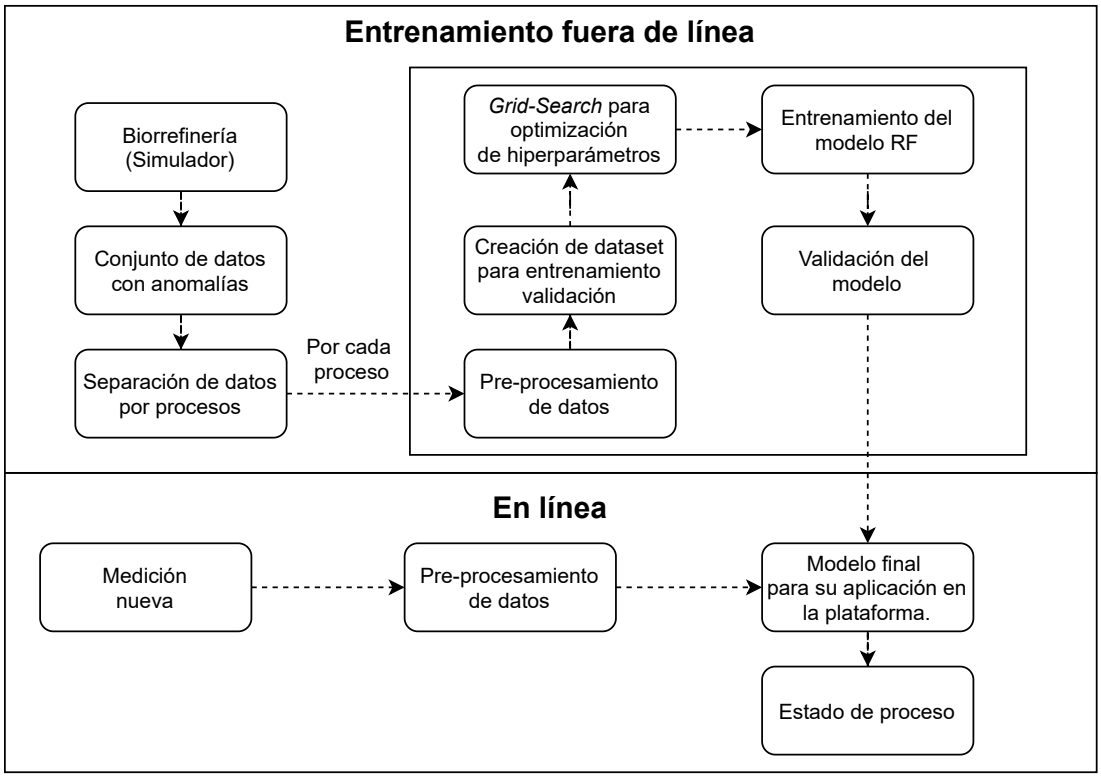

Figura 3.4: Proceso de entrenamiento

3.3.3 Preparación de los datos

Previo al desarrollo de cualquier modelo usando técnicas de *data-driven*, el primer paso fundamental es extraer información relevante del conjunto de datos. A menudo se realiza un pre-procesamiento de dichos datos para obtener un conjunto de entrenamiento. El conjunto de entrenamiento contiene datos del funcionamiento de los procesos fuera de línea y se utiliza para desarrollar las medidas que definen las operaciones bajo control, y las operaciones anómalas. Los procedimientos de pre-procesamiento suelen incluir [99]: eliminación de variables, auto escalamiento de variables y manejo de datos atípicos.

En ocasiones el conjunto de entrenamiento puede contener variables no relevantes para el monitoreo de un proceso. Por ejemplo, se puede saber a priori que ciertas variables exhiben errores de medición extremadamente grandes, debido a una mal calibración de algún sensor, o algunas de las variables pueden estar físicamente separadas de la parte del proceso que se está monitoreando [33]. Es recomendable eliminar estás variables antes de un análisis posterior. *Principal Component Analysis* es una de las herramientas más utilizadas para realizar esta tarea; sin embargo, existe un campo bastante extenso dedicado al estudio de métodos para selección y eliminación de variables.

El auto escalamiento estandariza las variables del proceso de una manera que garantiza que cada variable se encuentre dentro de un mismo rango, asegurando que las variables con altas variaciones no dominen [33]. El primer paso es restar a cada variable su media muestral. El segundo paso es dividir cada variable, ya con la media centrada, por su desviación estándar. Como paso adicional, posterior al entrenamiento del modelo, se debe realizar el auto escalado usando la media y desviación estándar usadas en el conjunto de entrenamiento para cada nueva medición que sea candidato a ser evaluado por el modelo entrenado.

Por último, los valores atípicos, como se ha descrito previamente, son valores de medición aislados que son erróneos, por ejemplo, valores nulos. Estos valores, en el contexto de un conjunto de entrenamiento, pueden influir significativamente en la estimación de parámetros estadísticos y otros parámetros relacionados con una medida dada. Eliminar los valores atípicos del conjunto de entrenamiento puede mejorar significativamente la estimación de los parámetros y debería ser un paso esencial al pre-procesar los datos [33]. Sin embargo, la eliminación debe tener una justificación valida, y preferentemente, esta debe ser validada por la persona experta en el tema.

Así, siguiendo los principios mencionados previamente, la estructura del conjunto de entrenamiento y de validación quedó de la siguiente manera. Primero, se tienen dos conjuntos de datos fuentes, uno del proceso DA y el otro del proceso MEC. Cada dato, en estos conjuntos, está etiquetado con 0 (Normal) o 1 (Anormal). La proporción es mayor para la clase 1 (Tabla 3.1), es decir, se tienen conjuntos de datos desequilibrado. Cada conjunto de datos fuente se dividió en un conjunto de entrenamiento y otro de prueba (para probar el modelo final). Además, cada conjunto de entrenamiento se subdividió en otro conjunto de entrenamiento y validación (para la elección de hiperparámetros).

Clase	Mediciones en A	Mediciones en B
Normal (0)	105403	123165
Anormal (1)	561813	544051

Tabla 3.1: Cantidad de datos normales y anómalos por cada proceso

3.3.4 Optimización de hiperparámetros y entrenamiento de los modelos

La búsqueda de los mejores hiperparámetros se hizo utilizando *GridSearch*. Esta técnica nos permite evaluar todas las combinaciones posibles de valores de hiperparámetros, mediante validación cruzada (*cross-validation*) y visualizar el desempeño de los modelos entrenados por estas combinaciones. Los parámetros probados fueron, *n_estimators* o número de árboles; *criterion* o función para medir la calidad de la división; *max_features* o número máximo de características a considerar en cada división; *max_depth* o profundidad máxima para los árboles de decisión; *max_samples_split* o el número mínimo de muestras requeridas

para dividir un nodo; *min_samples_leaf* o el número mínimo de muestras requeridas para cada nodo terminal; y *bootstrap* o método de selección de muestras. Es decir, si muestras aleatorias iniciales se usan al construir los árboles o si todo el conjunto de datos se usa para construir cada árbol. En el Código 3.1 se detallan los valores probados.

Código 3.1: Paramétros probados con *Grid-Search*

```
{
'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
'criterion': ['gini', 'entropy'],
'max_features': ['auto', 'sqrt'],
'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'bootstrap': [True, False]}
```

Con los resultados obtenidos de *Grid-Search*, entrenamos los modelos *Random Forest* con los siguientes parámetros:

Código 3.2: Paramétros seleccionados para DA

```
{'bootstrap': True,
'criterion': 'gini',
'max_depth': 10,
'max_features': 'auto',
'min_samples_leaf': 1,
'min_samples_split': 5,
'n_estimators': 20}
```

Código 3.3: Paramétros seleccionados para MEC

```
{'bootstrap': True,
'criterion': 'gini',
'max_depth': 10,
'max_features': 'auto',
'min_samples_leaf': 1,
'min_samples_split': 2,
'n_estimators': 10}
```

3.3.5 Validación del modelo

Los modelos finales se han validado utilizando *K-Fold Cross-Validation*. Ésta es una técnica frecuentemente usada para la validación de modelos en aprendizaje automático. Básicamente, se divide aleatoriamente el conjunto de entrenamiento en k subconjuntos distintos llamados pliegues (*folds*). Luego, se entrena y evalúa el modelo K veces, eligiendo un pliegue diferente para la evaluación. El resultado es una matriz que contiene k puntajes de evaluación (Figura 3.5).

Las métricas utilizadas para la validación fueron:

- *Accuracy*: describe la relación entre el número de predicciones correctas y el número total de muestras de entrada.
- *Precision*: fracción de predicciones positivas que en realidad son positivas.

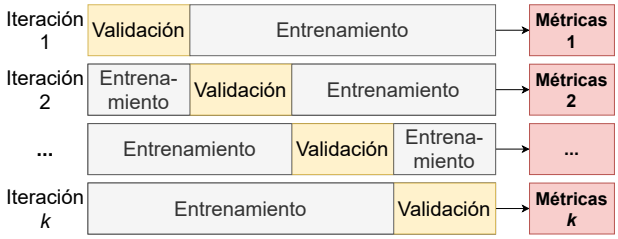

Figura 3.5: Kfold Cross-Validation

- *Recall*: captura la fracción de datos positivos identificados correctamente por el modelo.
- *F1-Score*: es la media armónica entre *precision* y *recall*. Su rango de valor es [0, 1]. Indica qué tan preciso es el modelo de clasificación (cuántas instancias clasifica correctamente), así como qué tan robusto es (no pierde un número significativo de instancias).

3.3.6 Resultados

DA		**MEC**	
Métrica	Promedio	Métrica	Promedio
accuracy	0.9997	accuracy	0.9998
precision	0.9999	precision	0.9999
recall	0.9997	recall	0.9998
f1_score	0.9998	f1_score	0.9999

Tabla 3.2: Rendimiento de los modelos por cada proceso usando 10-fold cross-validation

Los resultados del análisis (Tablas 3.2, 3.3 y 3.4) muestran que los modelos entrenados usando un clasificador *Random Forest* brindan puntuaciones consistentes y prometedoras en todas las métricas definidas. Tanto los altos valores de *precision* como los de *recall*, indican un buen rendimiento de los modelos. Esto es lo que se requiere para nuestro propósito previsto, es decir, identificar con gran confianza una fracción significativa de los estados anormales en los procesos. También tiene la ventaja de que puede interpretarse relativamente bien para los expertos. Por ejemplo, podemos extraer los arboles de decisión y explorar los caminos de decisión.

Elegimos *Random Forest* como el clasificador que se implementará en el prototipo, teniendo en cuenta que los datos provienen de un simulador y que *Random Forest* no necesariamente encajaría con los datos reales.

		Predicho									
		Normal	Anormal	Normal	Anormal	Normal	Anormal	Normal	Anormal	Normal	Anormal
		1-Fold		**2-Fold**		**3-Fold**		**4-Fold**		**5-Fold**	
R	Normal	7419	1	7515	0	7325	0	7300	1	7349	4
	Anormal	8	39278	1	39189	13	39367	1	39403	17	39335
e		**6-Fold**		**7-Fold**		**8-Fold**		**9-Fold**		**10-Fold**	
a	Normal	7237	0	7327	0	7373	1	7337	0	7370	0
l	Anormal	3	39465	1	39377	2	39329	11	39357	1	39334

Tabla 3.3: Matrices de confusión para 10-fold Cross-Validation del proceso DA

		Predicho									
		Normal	Anormal	Normal	Anormal	Normal	Anormal	Normal	Anormal	Normal	Anormal
		1-Fold		**2-Fold**		**3-Fold**		**4-Fold**		**5-Fold**	
R	Normal	8645	0	8622	1	8588	2	8524	1	8608	2
	Anormal	4	38057	2	38080	1	38114	2	38178	2	38093
e		**6-Fold**		**7-Fold**		**8-Fold**		**9-Fold**		**10-Fold**	
a	Normal	8595	2	8684	0	8680	0	8576	1	8578	3
l	Anormal	2	38106	4	38017	20	38005	2	38126	8	38116

Tabla 3.4: Matrices de confusión para 10-fold Cross-Validation del proceso MEC

Capítulo 4

Sistema de recomendaciones basado en el conocimiento

En el capítulo anterior construimos dos modelos guiados por datos que clasifican estados para los procesos de la biorrefinería de interés. Los modelos se construyeron sin profundizar en la estructura o funcionamiento de la biorrefinería, simplemente utilizando los datos provenientes del simulador. Ahora supongamos que la biorrefinería es un mundo, y que queremos comprender este mundo, o una parte de él. Por ejemplo, cómo funcionan los procesos de la biorrefinería, porqué se quiere emitir una recomendación a los operadores o por qué se quiere saber cómo se comportará el mundo si se cumplen ciertas condiciones (por ejemplo, le preocupa conocer qué variable se ve afectada por otra que está funcionando mal). El campo de la representación del conocimiento (*Knowledge representation o KR*) tiene como objetivo proporcionar bases informáticas para responder precisamente a este tipo de cuestionamientos. En este capítulo trataremos los conceptos básicos de los sistemas basados en el conocimiento. Así mismo, se abordará la creación de un modelo ontológico de la biorrefinería de interés. El modelo propuesto será utilizado para detectar estados anómalos y emitir recomendaciones.

4.1 Sistemas basados en conocimiento

La representación del conocimiento como método de modelado estudia la formalización del conocimiento y su procesamiento dentro de las computadoras. En términos muy generales, se trata de modelar simbólicamente el conocimiento humano y el razonamiento de tal manera que este conocimiento codificado pueda ser procesado por una computadora para obtener un comportamiento inteligente [100]. Los sistemas construidos sobre tales modelos se denominan sistemas basados en el conocimiento y a sus representaciones simbólicas involucradas se les conoce como bases de conocimiento (KB, por sus siglas en inglés). Las aplicaciones de KR abarcan desde tecnologías semánticas, aprendizaje automático e inteligencia artificial, integración de información, interoperabilidad de datos y comprensión del lenguaje natural.

Los sistemas basados en el conocimiento tienen un modelo computacional de algún dominio de interés en el que los símbolos sirven como sustitutos de los artefactos del dominio en el mundo real, como objetos físicos, eventos, relaciones, etc. [101]. El dominio de interés puede abarcar cualquier parte del mundo real o cualquier sistema hipotético sobre el que se desee representar el conocimiento con fines computacionales. Por otro lado, los componentes principales de un sistema basado en conocimiento son su base de conocimiento y un motor de razonamiento [102]. Básicamente, el sistema mantiene una base de conocimiento que almacena los símbolos del modelo computacional en forma de declaraciones sobre el dominio, y realiza el razonamiento mediante la manipulación de estos símbolos. Utiliza técnicas de razonamiento automatizado que permiten que un sistema informático saque conclusiones del conocimiento representado en una forma interpretable por la computadora [59]. Esto implica el uso de conocimiento y diferentes procesos cognitivos, que juegan una función entrelazada para facilitar el desarrollo de la asociación entre nuevos conceptos y los conceptos previamente adquiridos [103]. Así, las aplicaciones pueden basar sus decisiones en preguntas relevantes para el dominio planteadas sobre una base de conocimiento.

4.1.1 Base de conocimiento

El conocimiento se trata de información aplicable, es decir, es información contextualizada en un determinado dominio y, por lo tanto, cualquier conocimiento se relaciona con más conocimiento de una manera particular y diferente en cada individuo [104]. Por otro lado, el conocimiento, como se presenta generalmente, aparece principalmente de manera no estructurada o no de manera uniforme, lo que lo hace inadecuado para su posterior procesamiento [105]. Por ejemplo, las personas comúnmente usan conceptos que no tienen una definición única o un acuerdo en común de lo que significa el concepto. Sin embargo, a través de la observación entendemos cómo se utilizan cada concepto en la comunicación humana para crear un significado específico, más o menos definido, por ejemplo: un padre, una mascota, un vehículo o alguna idea más abstracta.

Entonces, lo que hace que un sistema se base en el conocimiento, no es el uso de un formalismo lógico, o el hecho de que es lo suficientemente complejo como para merecer una descripción intencional que implica conocimiento, más bien, es la presencia de una base de conocimiento, una colección de estructuras simbólicas que representan lo que cree y razona durante la operación del sistema [102]. Ejemplos de tales declaraciones en el dominio del funcionamiento de la biorrefinería de interés son, “un proceso tiene variables” o “cada variable es medido por un sensor”. Este conocimiento se puede utilizar para responder preguntas sobre el dominio de interés. A partir de las declaraciones dadas, y mediante una deducción automática, un sistema basado en el conocimiento puede deducir que “un proceso cuyas variables de entrada son: tasa de dilución, demanda química de oxígeno y ácidos grasos volátiles es un digestor anaerobio”.

Por consiguiente, podemos definir a una base de conocimiento (KB, por sus siglas en inglés) como la representación simbólica de conocimiento sobre un dominio de aplicación. Usamos la expresión “dominio de aplicación” para denotar la parte del mundo (que puede ser real o ficticio, un modelo sofisticado de un sistema o un modelo de competencia

experta) sobre la cual representamos conocimiento y razonamiento. Una KB generalmente contiene diferentes tipos de conocimiento, típicamente una ontología, hechos, reglas y restricciones [102].

4.1.2 Motor de razonamiento

En general, el razonamiento es la manipulación formal de los símbolos que representan una colección de proposiciones creídas para producir nuevas representaciones de proposiciones [100]. Una analogía útil es pensar en la suma binaria como una cierta manipulación formal. Por ejemplo, comenzamos con los símbolos "101" y "10", y terminamos con "0111". La manipulación en este caso es la suma binaria, y la nueva representación obtenida de esta manipulación es el símbolo "0111". El razonamiento es similar: podríamos comenzar con las oraciones "Un digestor anaerobio produce ácidos grasos volátiles" y "los ácidos grasos volátiles son entrada para las celdas de electrolisis microbianas", y después de una cierta cantidad de manipulaciones producir la oración: "el producto de un digestor anaerobio es la entrada para las celdas de electrolisis microbianas". A esta forma de razonamiento la llamaremos *inferencia lógica* [102], porque la oración final representa una conclusión lógica de las proposiciones representadas por las iniciales, como veremos más adelante. Así, el razonamiento es una forma de cálculo, no muy diferente de la aritmética, solo que realizada sobre símbolos que representan proposiciones en lugar de números [102].

Ahora, retomando el tema de sistemas basados en el conocimiento, el segundo elemento importante es el motor de razonamiento. El motor de razonamiento procesa el conocimiento en una KB para responder alguna pregunta o resolver algún objetivo [101]. El motor utiliza algoritmos que procesan los elementos de la base de conocimiento para construir "nuevos" conocimientos, es decir, nuevas construcciones simbólicas que solo están implícitas en la base del conocimiento [101]. Es necesario enfatizar que en un sistema basado en el conocimiento no puede tener una representación del conocimiento sin tener mecanismos de razonamiento [96].

Usando un motor de razonamiento semántico en una ontología podemos [106]:

- Consistencia: Comprobar si el conocimiento es significativo en una ontología.
- Subsunción: Estructurar el conocimiento y computarizar la taxonomía.
- Equivalencia: Comprobar si dos clases denotan el mismo conjunto de instancias.
- Instanciación: Comprobar si el individuo es una instancia de una clase.
- Recuperación: Recuperar el conjunto de individuos que instancian una clase y una propiedad

4.2 Ontologías

Una ontología permite describir los objetos o conceptos en un dominio, generalmente pero no siempre de una manera similar a una taxonomía, con la intención de obtener una

descripción formal de la terminología en un dominio [101]. El término ontología, tiene definiciones desde muchos puntos de vista diferentes y con distintos grados de formalidad. Entre varias definiciones propuestas que explican lo que es una ontología en el contexto de la informática, resaltamos la siguiente [107]:

"Una ontología es una especificación formal y explícita de una conceptualización compartida."

En esta definición hay que destacar tres aspectos importantes de las ontologías: las ontologías son conceptualizaciones; son explícitas; y se comparten.

Una "conceptualización" se refiere a modelos abstractos que consisten en conceptos que son relevantes para describir el mundo real [107]. Es decir, las ontologías actúan como una especie de sustituto de la realidad. Que sean "explícitos" significa que los tipos de conceptos utilizados y las restricciones sobre su uso se definen explícitamente [107]. Por ejemplo, en dominios médicos, los conceptos son enfermedades y síntomas, las relaciones entre ellos son causales y una restricción es que una enfermedad no puede causarse a sí misma. "Formal" se refiere al hecho de que la ontología debe ser legible por computadoras [107]. El uso del lenguaje natural no es apropiado debido a su ambigüedad, inconsistencia y especificación incompleta. "Compartido" refleja la noción de que una ontología captura conocimiento consensuado, es decir, no es privado para un individuo, sino que es aceptado por un grupo [107].

La ontología generalmente se construye sobre una taxonomía: una estructura jerárquica de conceptos que limita la relación entre los conceptos con la formulación "es un tipo de" [107]. La taxonomía es construida al agregar una red más rica de relaciones semánticas y componentes adicionales como funciones, relaciones o restricciones, reglas de inferencia y axiomas. De este modo, el valor de las ontologías radica especialmente en el hecho de que están destinadas a organizar conceptos bajo una especificación común, que a menudo cubre un dominio completo, para facilitar el intercambio de conocimientos [105]. Por lo tanto, las ontologías son el medio más viable para realizar una representación de conocimiento debido a sus beneficios y características.

4.2.1 Lenguajes ontológicos y OWL

Las ontologías se codifican comúnmente mediante el uso de lenguajes formales que se denominan lenguajes ontológicos [105]. Los lenguajes ontológicos permiten compartir conocimientos y al mismo tiempo respaldan su procesamiento al proporcionar reglas de razonamiento. En términos generales, los lenguajes ontológicos se pueden clasificar en dos grupos dependiendo del sistema de lógica en el que se base.

Lenguajes de lógicas de primer orden

El primer grupo de lenguajes ontológicos son los basados en *lógicas de primer orden*. La lógica de primer orden (FOL, por sus siglas en inglés) es superior a la lógica proposicional, ya que permite variables y cuantificadores [100]. FOL nos permite decir "x es un proceso", donde tratamos al sujeto como una variable y convertimos "es un proceso" en un predicado.

Podemos establecer relaciones entre estos predicados con conectores lógicos, como las declaraciones *AND*, *OR*, *NOT* e *IF-THEN*. Además, permiten aplicar cuantificadores a los enunciados utilizando cuantificadores universales ("para cada") y existenciales ("existe"), así como utilizar el concepto de negación. Ejemplos de estos lenguajes incluyen CYCL, KIF y Common Logic.

Lenguajes de lógicas de descripción

Por otra parte, están los lenguajes ontológicos basados en *lógicas de descripción* (DL, por sus siglas en inglés) que se originan tradicionalmente en el campo de la inteligencia artificial. Ejemplos de estos lenguajes son KL-ONE y LOOM. KL-ONE introdujo la mayoría de las nociones clave exploradas en el extenso trabajo sobre las lógicas de descripción que surgieron posteriormente. Por ejemplo, las nociones de concepto y roles y cómo debían estar interrelacionados [8].

De forma general, en los lenguajes de DL, una base de conocimiento se divide en un componente terminológico, el TBox, que puede verse como la parte ontológica de la KB, y un componente de aserción, es decir, el ABox, que contiene afirmaciones sobre individuos [100] (Figura 4.1). Además, los lenguajes de DL usan una semántica modelo-teórica. Por lo tanto, las declaraciones en el TBox y en el ABox pueden identificarse con fórmulas en lógica de primer orden o, en algunos casos, una ligera extensión del mismo [8].

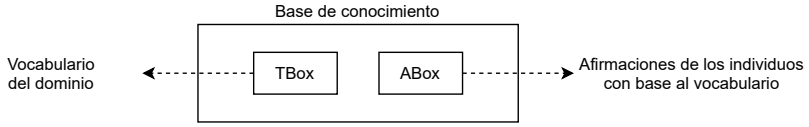

Figura 4.1: Componentes de una KB en DL

De este modo, los lenguajes DLs representan el conocimiento en términos de conceptos o conjuntos de objetos, y roles, que denotan relaciones binarias entre instancias de conceptos [101]. Así, partiendo de conceptos atómicos y roles atómicos, y utilizando un conjunto de constructores, se pueden construir conceptos y roles más complejos. Por lo tanto, los lenguajes de descripción se distinguen por los constructores que proporcionan. Por ejemplo, en la Tabla 4.1 se aprecia las reglas para construir conceptos en una familia de lenguajes $\mathcal{AL}$ (Lenguajes Atributivos) [8], y en la Tabla 4.2 puede observar abreviaciones para extender los lenguajes $\mathcal{AL}$.

Por último, en los últimos años ha surgido una nueva familia de lenguajes ontológicos con el propósito de compartir conocimientos en Internet [108]. Los ejemplos de estos lenguajes incluyen el lenguaje de intercambio de ontología (XOL) basado en XML, la extensión de ontología HTML simple (SHOE), el lenguaje de intercambio de ontología (OIL) y Web Ontology Language (OWL).

Sintaxis	Descripción
$C, D \rightarrow A\|$	Concepto atómico
$\top\|$	Concepto universal
$\bot\|$	Concepto inferior
$\neg A\|$	Negación de concepto atómico
$C \sqcap D\|$	Intersección
$\forall R.C\|$	Restricción de valor
$\exists R.\top\|$	Cuantificación existencial limitada

Tabla 4.1: Reglas de sintaxis para formación de conceptos en un lenguaje atributivo ($\mathcal{AL}$)

Símbolo	Descripción
$\mathcal{AL}$	Lenguaje Atributivo: • Negación atómica. • Intersección de conceptos. • Restricciones universales. • Cuantificación existencial. limitada
$\mathcal{FL}-$	Sublenguaje de $\mathcal{AL}$ sin negación atómica.
$\mathcal{FL}_o$	Sublenguaje de $\mathcal{FL}-$ sin limitación de cuantificación existencial.
$\mathcal{C}$	Negación de conceptos complejos .
$\mathcal{S}$	$\mathcal{AL}$ y $\mathcal{C}$ con propiedades transitivas.
$\mathcal{H}$	Jerarquía de roles. (subpropiedades)
$\mathcal{R}$	• Limitación de axiomas de inclusión de roles complejos. • Reflexividad e irreflexividad. • Disyunción de roles
$\mathcal{O}$	Nominales, es decir, clases enumeradas y restricciones de valor de objeto.
$\mathcal{I}$	Propiedades inversas
$\mathcal{N}$	Restricción de cardinalidad
$\mathcal{Q}$	Restricciones de cardinalidad calificadas
$\mathcal{F}$	Propiedades funcionales
$\mathcal{E}$	Cuantificación existencial completa
$\mathcal{U}$	Unión de conceptos
$(\mathcal{D})$	Propiedades de tipo de datos, valores de datos y tipos de datos

Tabla 4.2: Abreviaciones para DLs [8]

OWL

En noviembre de 2001, el *World Wide Web Consortium* (W3C) se propuso desarrollar un nuevo lenguaje de ontología web (OWL) que se utilizaría para la representación del conocimiento de forma que pudiera compartirse en la web [108]. Los primeros pasos hacia esta web semántica fue el de crear una especificación para una forma de agregar metadatos simples y relativamente livianos a los documentos en la web [108]. Como resultado de esto surgieron los lenguajes *Resource Description Framework* (RDF) y RDF Schema (RDFS), los cuales fueron utilizados para la representación de metadatos en la web. En este punto OWL está estrechamente relacionado con el desarrollo de la web semántica, cuyo propósito era el de mejorar la interpretación de las computadoras con respecto a la información almacenada en la web [108]. No solo para mejorar la accesibilidad y la comunicación de esta información a los humanos, sino principalmente para el intercambio de conocimientos entre servicios web automatizados.

OWL se convirtió en un miembro destacado de la familia de lenguajes de representación del conocimiento, diseñada específicamente para la representación de ontologías. Su expresividad está restringida para garantizar propiedades computacionales favorables [100]. Además, OWL extiende su sintaxis y semántica de los lenguajes RDF y RDFS, los cuales utilizan expresiones llamadas tripletas de la forma sujeto-predicado-objeto para hacer declaraciones sobre recursos [100]. Las ontologías OWL consisten, generalmente, en individuos o instancias, propiedades y clases [109].

Los diversos requisitos (por ejemplo, el nivel de expresividad) para un lenguaje ontológico web llevaron a la especificación de tres tipos, cada una enfatizando diferentes características: OWL Full, DL y Lite [100]. El primer tipo, OWL Full, es una extensión semántica de RDFS que agrega una serie de primitivas de representación de conocimiento que antes no estaban disponibles [109]. Esto lo convierte en el sub-lenguaje OWL más expresivo. Está destinado a ser utilizado en situaciones donde una alta expresividad es más importante que poder garantizar la capacidad de decisión o la integridad computacional del lenguaje. La ventaja de OWL Full es que es totalmente compatible con RDF, tanto sintáctica como semánticamente [100]: cualquier documento RDF válido también es un documento OWL completo válido, y cualquier conclusión válida RDF / RDFS también una conclusión completa OWL válida. La desventaja de OWL Full es que el lenguaje se ha vuelto tan robusto como para ser indecidible, eliminando cualquier esperanza de soporte de razonamiento completo (y mucho menos eficiente) [109].

El segunda tipo, OWL DL, es mucho más expresivo que OWL-Lite y se basa en lógicas de descripción (de ahí el sufijo DL) [100]. Es un subconjunto sintáctico de OWL Full, pero limita la semántica a la lógica de descripción $\mathcal{SHOIN}(\mathcal{D})$ [108]. Además, en su momento este lenguaje era la lógica de descripción mayormente expresiva para la cual se conocían algoritmos eficientes y existían casos de uso [108]. La ventaja de esto es que permite un soporte de razonamiento eficiente. La desventaja es que perdemos la compatibilidad total con RDF: un documento RDF en general tendrá que extenderse de alguna manera y restringirse en otros antes de que sea un documento OWL DL válido. Por el contrario, cada documento OWL DL válido sigue siendo un documento RDF válido.

Por último está OWL-Lite, cuya expresividad es menor a los dos mencionados previamente. Es un subconjunto de OWL DL, donde la semántica está diseñada para limitarse a la lógica de descripción $\mathcal{SHIF}(\mathcal{D})$ [108]. Está destinado a ser utilizado en situaciones donde solo se necesitan una jerarquía de clases simple y restricciones simples. La ventaja de esto es un lenguaje que es más fácil de entender (para los usuarios) y más fácil de implementar (para los creadores de herramientas). La desventaja es, por supuesto, una expresividad restringida.

4.3 Detección de anomalías utilizando representación del conocimiento

A diferencia del enfoque de *data-driven* explorado en el Capítulo 3 que crea modelos utilizando datos de procesos, un enfoque basado en la representación del conocimiento utiliza modelos cualitativos. El núcleo de un sistema de detección basado en el conocimiento consiste en una base de conocimiento, una base de datos, un motor de inferencia y un componente de explicación [71]. Básicamente, la detección de anomalías basado en el conocimiento consiste en utilizar un modelo cualitativo que representa el conocimiento a priori del proceso bajo monitoreo [110]. La detección se realiza ejecutando reglas y algoritmos de búsqueda bien definidos, donde la idea básica es detectar anomalías entre las actividades esperadas y observadas. Las principales ventajas de estos sistemas son su facilidad de desarrollo, razonamiento transparente, la capacidad de razonar bajo incertidumbre y la capacidad de proporcionar explicaciones de eventos [111].

Los enfoques basados en el conocimiento se han estudiado ampliamente en la literatura, y hay varias trabajos de investigación disponibles. Por ejemplo, en [110] presentan un enfoque para la automatización de agentes utilizando una ontología para formalizar el conocimiento del agente. El modelo obtenido es una representación explícita del entorno externo y los elementos internos del agente, y se utiliza para detectar anomalías en su propio comportamiento. Por otra parte en [112] se presenta la estructura y el modo de uso de un sistema integrado de diagnóstico y control de anomalías para equipos de sistemas de energía. El sistema de diagnóstico y control permite el análisis en línea mediante aplicación de escritorio, una aplicación web y un análisis fuera de línea para determinar las fallas del transformador. Además, brinda soluciones basadas en ciertos síntomas observados en el equipo y su comparación con los resultados de la encuesta del estado del sistema. A su vez, el enfoque presenta un sistema experto basado en web con una base de conocimiento y conectividad a bases de datos. El propósito de este trabajo es desarrollar un shell de un sistema experto basado en web y luego usarlo para el diagnóstico y control de anomalías. Por último, en [113] proponen un método de diagnóstico de fallas basado en ontologías para vehículos de carga pesada. Se propone el uso de un modelo ontológico para lograr la integración, el intercambio y la reutilización del conocimiento del sistema. Además, combinado con la ontología, se introduce CBR (razonamiento basado en casos) para realizar diagnósticos efectivos y precisos de anomalías siguiendo cuatro pasos (selección de características, recuperación de casos, coincidencia de casos y actualización de casos);

también para cubrir la escasez del método CBR debido a la falta de casos en cuestión, RBR (razonamiento basado en reglas) basado en ontología se presenta a través de la construcción de reglas SWRL (*Semantic Web Rule Language*).

4.4 Modelo ontológico de la biorrefinería de interés

El objetivo principal de esta sección es el de presentar el diseño de un sistema de detección de anomalías para la biorrefinería de interés usando representación del conocimiento. Para alcanzar este objetivo se eligió a OWL-DL como lenguaje ontológico, por su máxima expresividad sin perder la integridad computacional (se garantiza que se computarán todas las implicaciones) y la capacidad de decisión (todos los cálculos terminarán en un tiempo finito) de los sistemas de razonamiento. Además, como framework de trabajo se eligió Protégé, un editor de ontologías de código abierto y gratuito para construir sistemas inteligentes.

Así, durante esta sección se introducirá el modelo ontológico para representar la información de los procesos que conforman esta biorrefinería, además del sistema de detección que facilita la obtención de un diagnóstico. Sin embargo, debemos recalcar que esta biorrefinería físicamente no existe y está en estado de propuesta, por lo que no se cuenta con mucha información de los procesos ni de la complejidad que los rige. Dicho esto, la base de conocimiento estará, en su mayoría, conformado por conceptos provenientes del simulador descrito en el Capítulo 2.

Como primer paso se modelará la base de conocimiento que incluirá conceptos como Proceso, Sensor, Variable, etc. De igual forma, las relaciones entre ellos (Proceso-Variable, Variable-Sensor). Posteriormente, para realizar un diagnóstico se definirán reglas SWRL, y finalmente, el sistema de recomendaciones a través de la relación Error-Recomendación.

4.4.1 Clases

Las clases OWL son conjuntos que contienen individuos. Se describen mediante descripciones formales que establecen con precisión los requisitos para ser miembro de la clase [114]. Las clases se pueden organizar en una jerarquía de superclase-subclase, que también se conoce como taxonomía. Las subclases son especializaciones de la superclases. Por ejemplo, considere las clases Animal y Gato: Gato podría ser una subclase de Animal (por lo que Animal es la superclase de Gato).

De esta forma, el primer paso consistió en la adquisición de información sobre el dominio de interés. Esto se realizó a través de dos fuentes; primero, realizando entrevistas con el experto y segundo, haciendo un análisis del simulador. La información recabada en el dominio de un sistema de detección de anomalías para la biorrefinería planteada, se puede apreciar en la Tabla 4.3. Posteriormente, se siguió un proceso de análisis para esta información, teniendo como resultado la identificación de ocho clases principales y seis subclases (Figura 4.2).

Entidad	Descripción
Biorrefinería	Representa la abstracción de una planta que produce hidrógeno a partir de residuos agroindustriales. Está conformado por tres procesos: DA y MEC.
Proceso	Representa un proceso industrial que, por ejemplo, está relacionado con la generación de energía, la transmisión de energía, la distribución de energía, la industria química, o la automatización. Siendo específicos, en el dominio de interés se han identificado tres procesos: *digestión anaerobia*, *celdas de electrolisis microbianas* y *fotobiorreactor*. Se sabe que la producción de CO_2 y AGV del proceso DA, es un entrada para el proceso MEC.
Variable	Representa una magnitud que influye en el estado de un proceso y que, agrupaciones de esta entidad, forman un proceso industrial. Entre sus características tenemos que una variable debe tener un valor máximo y mínimo de operación, además de contar con una unidad de medida. Por otra parte, se han definido dos tipos de variables, de entrada y de salida (también llamadas variables de estado). A su vez, las variables de entrada pueden ser controladas o desconocidas; mientras que las variables de salida pueden ser medible o no medible.
Sensor	Representa una entidad usada para la medición o estimación de la magnitud de una variable. Dependiendo del tipo de variable a censar los sensores pueden ser: de hardware o software. Un sensor de hardware mide variables de entrada desconocida y variables de estado medibles. Por otro lado, un sensor de software estima el valor de variables de entrada desconocida y variables de estado no medibles. Una variable controlada no se mide a través de un sensor, sino que su valor es ya conocido o propuesto por los operadores. Cada sensor registra una lectura en un tiempo t de una variable.
Lectura	Representa la medición o estimación hecha por un sensor a una variable. Las lecturas tienen un estado con base a los limites de la variable que se está midiendo
Estado	Representa la condición actual de la entidad asociada.
Error	Representa una condición anormal de la entidad asociada. Un error puede afectar a una o más variables. Además, un error está asociado a una lectura, es decir, dependiendo de la lectura de un sensor se puede determinar la existencia de un error.
Recomendación	Representa una propuesta de solución para el error asociado.

Tabla 4.3: Descripción de entidades identificadas en el dominio de interés

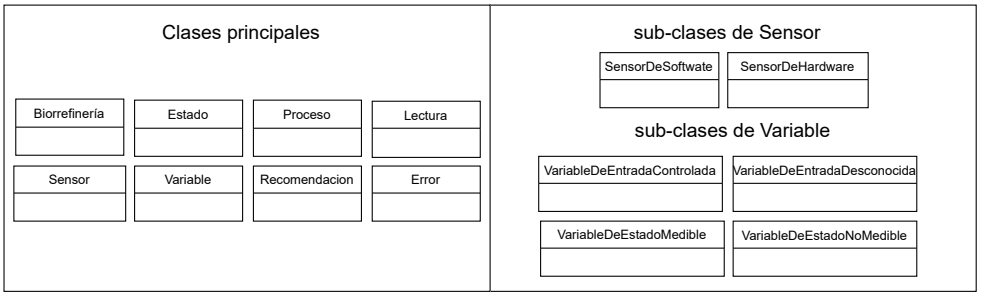

Figura 4.2: Clases identificadas en el dominio de la biorrefinería de interés

4.4.2 Propiedades

Las propiedades son relaciones binarias entre dos individuos (propiedades de objeto) o entre un individuo y un tipo de dato (propiedades de datos) [114]. Por ejemplo, la propiedad *tieneProceso* podría vincular un individuo de la clase *Biorrefinería* con individuos de la clase *Proceso*. Por otro lado, las propiedades pueden ser inversas, funcionales, transitivas o simétricas. Para mayor información consulte [114].

Siguiendo con el análisis de nuestro dominio de interés (Tabla 4.3), podemos identificar por ejemplo, que una biorrefinería tiene procesos, que un proceso está formado de variables o que un sensor censa una variable. Además, profundizando en las características de cada clase, podemos decir por ejemplo, que una variable tiene un nombre, una descripción, un valor máximo y un valor mínimo. En la Tabla 4.4 y 4.5 se enlistan tanto las propiedades de objetos como las de datos para el dominio de interés.

No.	Propiedad	Dominio	Rango	Inversa
1	afectaVariable	Error	Variable	N/A
2	alimentaProceso	Proceso	Proceso	esAlimentadoPor
3	censaVarible	Sensor	Variable	esCensadoPor
4	estimaVariable	SensorDeSoftware	Var...Desconocida or Var...NoMedible	esEstimadoPor
5	mideVariable	SensorDeHardware	Var...Desconocida or Var...Medible	esMedidoPor
6	enRiesgoDePresentar	Proceso	Error	N/A
7	esAlimentadoPor	Proceso	Proceso	N/A
8	esCensadoPor	Variable	Sensor	censaVarible
9	esEstimadoPor	Var...Desconocida or Var...NoMedible	SensorDeSoftware	estimaVariable
10	esMedidoPor	Var...Desconocida or Var...Medible	Sen...Hardware	mideVariable
11	esEstadoDe	Estado	Lectura or Proceso or Variable	tienEstado
12	esLecturaDe	Lectura	Sensor	tieneLectura
13	esProcesoDe	Proceso	Biorrefineria	tieneProceso
14	esRecomendacionDe	Recomendacion	Error	tieneRecomendacion
15	esUsadoParaEstimarPor	Lectura or Var...Controlada	SensorDeSoftware	utilizaParaEstimar
16	esVariableDe	Variable	Proceso	tieneVariable
17	presentaError	Proceso	Error	N/A
18	tieneEstado	Proceso	Lectura or Proceso or Variable	esEstadoDe
19	tieneLectura	Sensor	Lectura	esLecturaDe
20	tieneLecturaAnomala	Error	Lectura	N/A
21	tieneProceso	Biorrefineria	Proceso	esProcesoDe
22	tieneRecomendacion	Error	Recomendacion	esRecomendacionDe
23	tieneVariable	Proceso	Variable	esVariableDe
24	utilizaParaEstimar	SensorDeSoftware	Lectura or Var...Controlada	esUsadoParaEstimarPor

Tabla 4.4: Propiedades de objetos. Si una propiedad dada tiene una clase como dominio, significa que cualquier individuo que tenga un valor para la propiedad (es decir, es el sujeto de una relación a lo largo de la propiedad), se deducirá que es una instancia de la clase definida en dominio. Esto se aplica también para las clases declaradas en Rango. La propiedad de inversa hace referencia a que si existe un vínculo entre un individuo a con un individuo b, entonces su propiedad inversa vinculará al individuo b con el individuo a

No.	Propiedad	Dominio	Rango
1	esErrorDe	Error	string
2	tieneCategoria	Error	string
3	tieneDescripcion	Error or Proceso or Recomendacion or Variable	string
4	tieneDescripcionDeEstado	Proceso	string
5	tieneEstampaDeTiempo	Error or Lectura	string
6	tieneFechaDeActualizacion	Error or Proceso or Recomendacion or Variable	string
7	tieneFlujoDeDato	Sensor	string
8	tieneNivelDePeligro	Error	string
9	tieneNombre	Biorrefineria or Error or Proceso or Recomendacion or Sensor or Variable	string
10	tieneTipoDeLectura	Lectura	string
11	tieneUnidadDeMedida	Variable	string
12	tieneValorCensado	Lectura	float
13	tieneValorMaximo	Sensor or Variable	float
14	tieneValorMinimo	Sensor or Variable	float
15	tieneValorPropuesto	VariableDeEntradaControlada	float
16	tieneValorRiesgoMaximo	Variable	float
17	tieneValorRiesgoMinimo	Variable	float

Tabla 4.5: Propiedades de datos

De esta forma, traduciendo toda la información nueva a una sintaxis DL podemos tener la descripción de los conceptos para nuestro sistema. Por ejemplo, una Biorrefinería está constituida por al menos un Proceso y tiene un nombre que es de tipo *string*. Así, usando una sintaxis DL, la clase Biorrefinería se describe como:

$$
\begin{gathered}
Biorrefineria \equiv \top \sqcap \\
(\exists\, tieneProceso.Proceso) \sqcap \\
(= 1\ tieneNombre.string) \sqcap \\
(Sensor \sqsubseteq \bot) \sqcap \\
(Error \sqsubseteq \bot) \sqcap \\
(Proceso \sqsubseteq \bot) \sqcap \\
(Estado \sqsubseteq \bot) \sqcap \\
(Variable \sqsubseteq \bot) \sqcap \\
(Recomendacion \sqsubseteq \bot) \sqcap \\
(Lectura \sqsubseteq \bot)
\end{gathered}
$$

Para consultar el resto las clases diríjase al Anexo B.

Por último, en la Figura 4.3 se muestra la estructura del sistema ya con las propiedades previamente mencionadas.

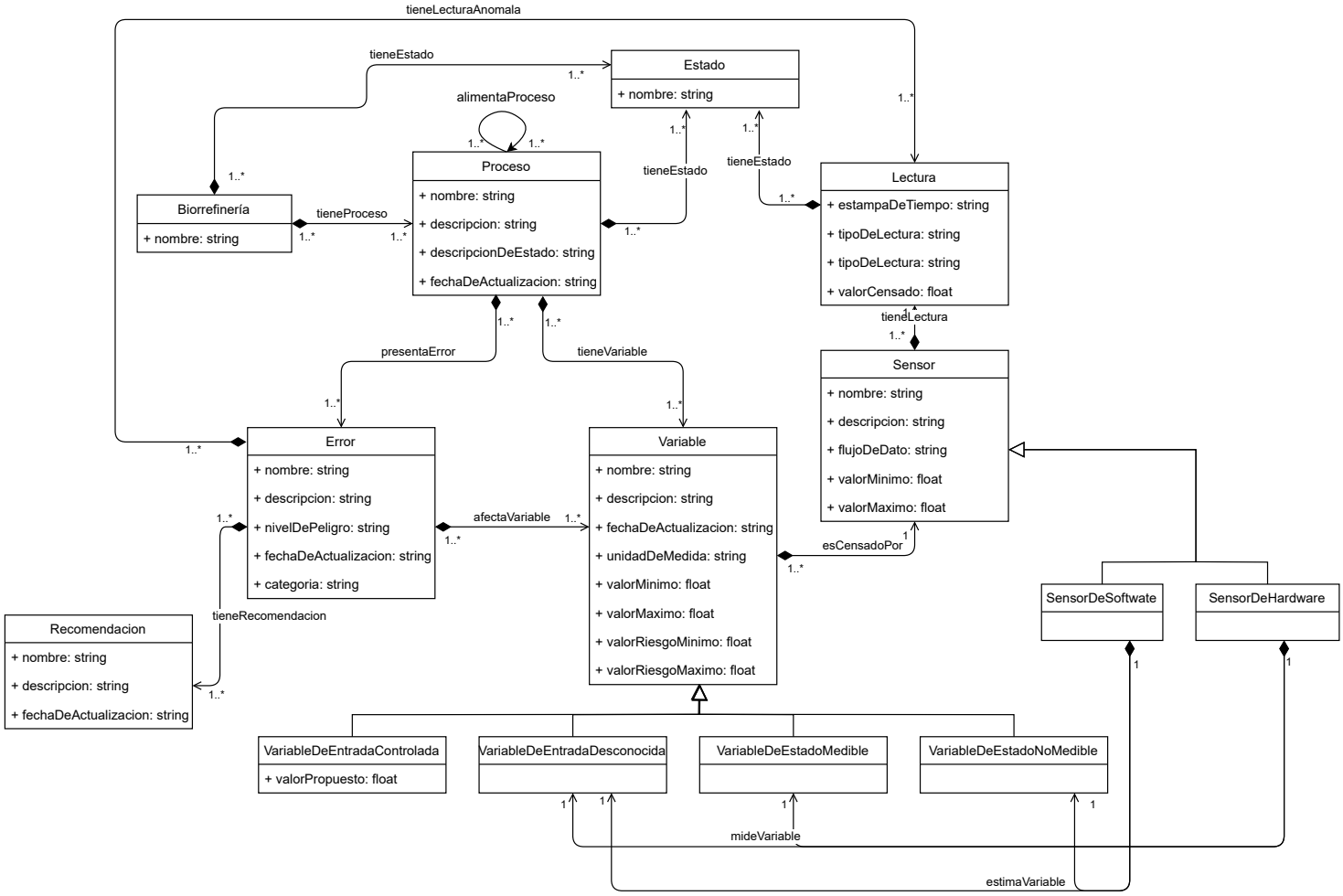

Figura 4.3: Diagrama de clases con propiedades y relaciones identificadas

4.4.3 Individuos y representaciones ontológicas del sistema

Los individuos, representan instancias de las clases de un dominio [114]. Al instanciar individuos podemos crear una representación específica de la taxonomía definida para nuestro dominio. De esta forma podemos simular, a través de la ontología, el funcionamiento de nuestro sistema. Por ejemplo, una biorrefinería de producción de hidrógeno con tres procesos principales basado en un simulador es representada de la siguiente forma:

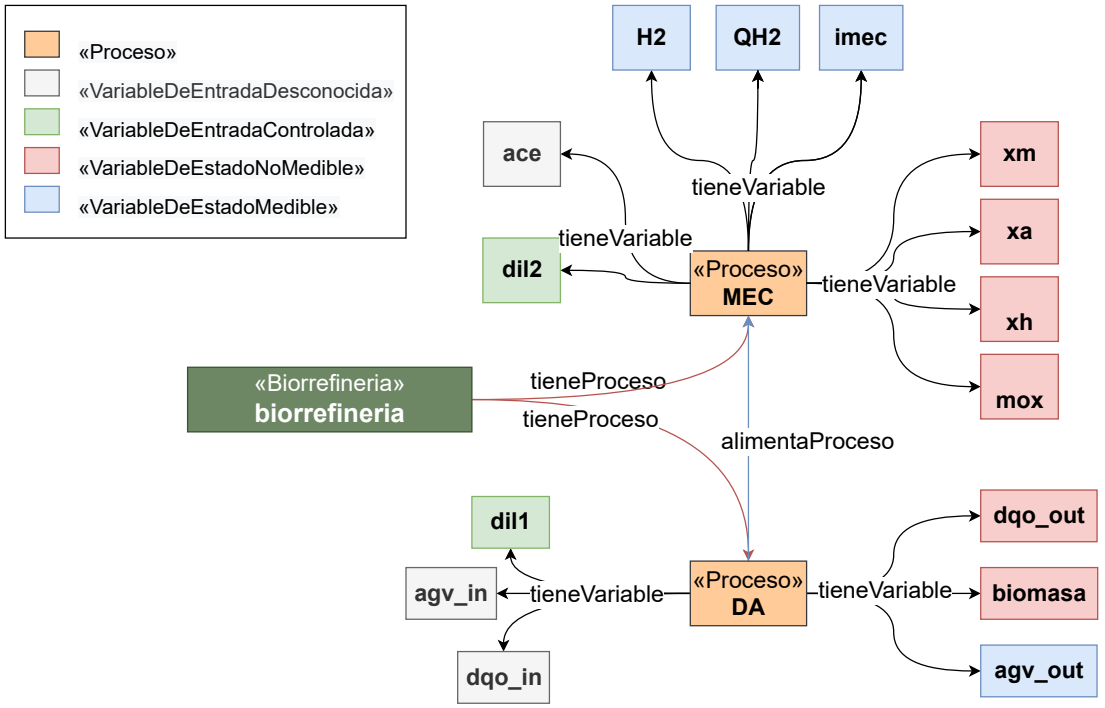

Figura 4.4: Biorrefinería de producción de hidrógeno con dos procesos: DA y MEC

Representación ontológica entre Variable, Sensor y Lectura

Los individuos de la clase Variable se relacionan con individuos de la clase Sensor a través de las propiedades de objeto "esMedidoPor" y "esEstimadoPor". A su vez, los individuos de Lectura guardan una relación con los individuos de la clase Sensor a través de la propiedad "tieneLectura" y "utilizaParaEstimar". En la Figura 4.5 se ejemplifica esta interacción para el proceso DA. Cabe señalar que no se han agregado todas las relaciones entre variables para facilitar su visualización.

Representación ontológica del sistema de detección de anomalías y emisión de recomendaciones

El sistema de detección de anomalías ontológico fue pensado para funcione con base a las lecturas de cada variable en un proceso. De esta forma, a través de reglas SWRL y utilizando los mínimos y máximos de cada variable como referencia, es posible detectar una falla dentro de un proceso. En la Figura 4.6 se puede visualizar el resultado que prosigue a la aplicación de las reglas SWRL cuando las lecturas de las variables están en los rangos permitidos.
Por otra parte, la Figura 4.7 presenta el escenario en donde las lecturas registradas no respetan los límites mínimos y máximos. En este caso, la lectura de la variable biomasa registró un valor mayor al máximo permitido. Esto provoca que el estado del proceso pase a *Falla* y, además, presente un *Error*. Este error puede afectar directa o indirectamente a otras

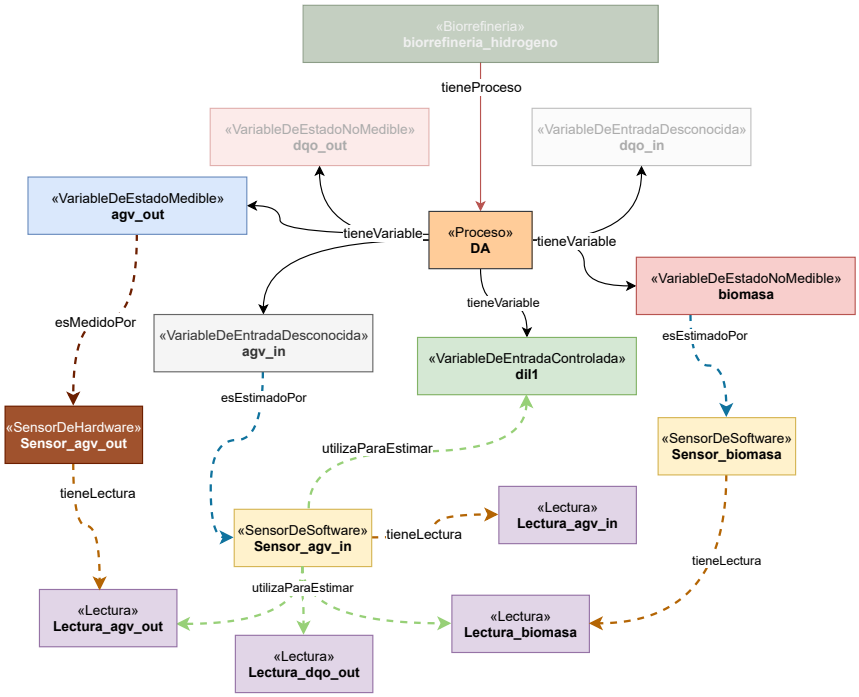

Figura 4.5: Interacción entre individuos de la clase Variable, Sensor y Lectura

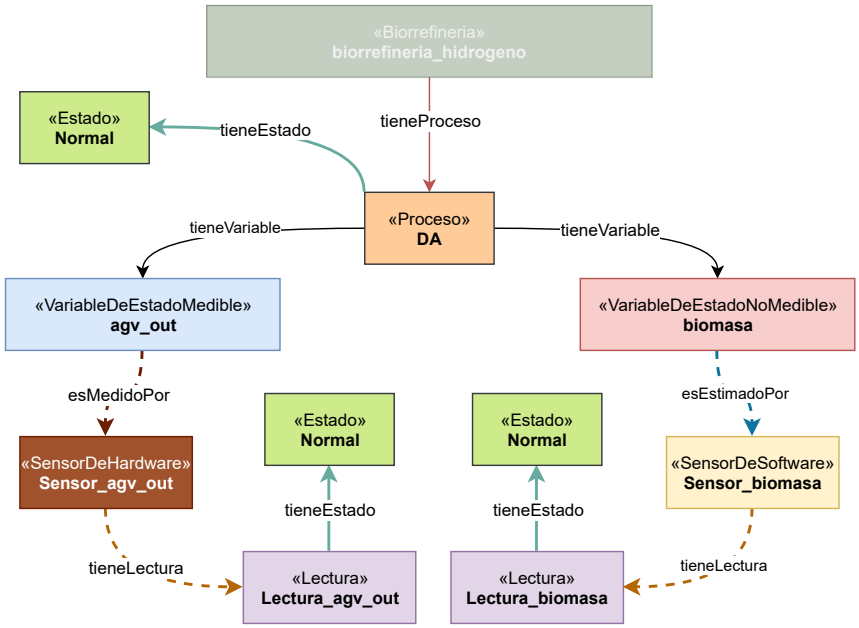

Figura 4.6: Sistema de detección de anomalías - Caso: Normal

variables, por ejemplo, el *Error_biomasa_alto* afecta directamente a la variable de estado *agv_out*, pero también, de forma indirecta, está afectando a las variables *agv_in* y *dqo_in*.

La afectación de estas dos variables se debe a que el sensor que mide cada una de ellas, utiliza la lectura de biomasa para estimar su valor. Aunque, para propósitos ilustrativos, esto solo se aprecia en la variable *agv_in*. Es importante mencionar que, aunque el error esté ejerciendo una influencia negativa en las variables afectadas, no necesariamente puede provocar que estas fallen. Por último, a cada individuo de la clase Error se le asocia un individuo de la clase Recomendación. El individuo recomendación tiene las indicaciones o consejos a seguir con tal de solventar o minimizar el error presentado.

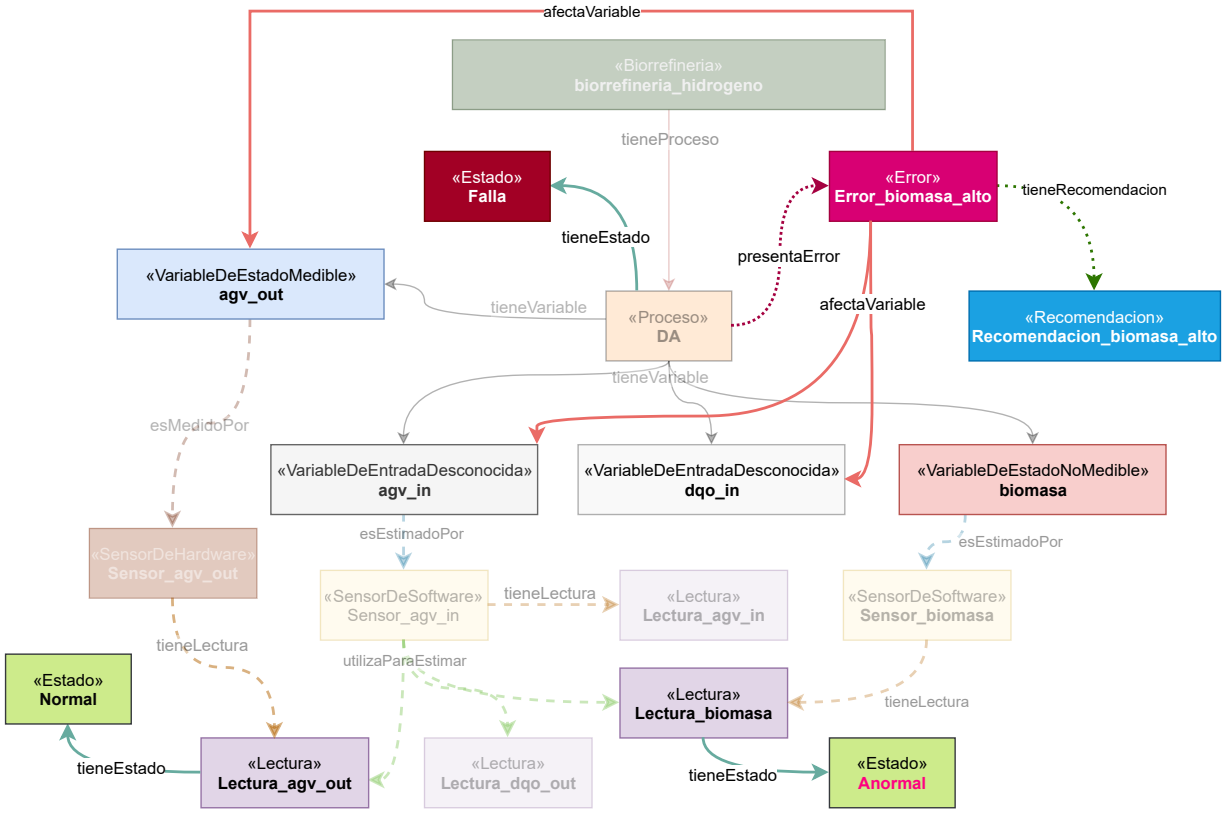

Figura 4.7: Sistema de detección de anomalías - Caso: Falla

4.4.4 Reglas SWRL

Semantic Web Rule Language o SWRL, es un lenguaje basado en la combinación de OWL-DL y sub-lenguajes de OWL-Lite [115]. Permite a los usuarios escribir reglas similares a las de Horn para razonar sobre los individuos OWL e inferir nuevos conocimientos de esos individuos [116]. Las reglas SWRL se definen en la forma de una implicación entre un antecedente (cuerpo o premisas) y el consecuente (cabeza) [116]. Una forma "legible" de una regla SWRL es:

$$antecedente \rightarrow consecuente$$

El significado deseado de las reglas SWRL puede leerse como: siempre que se cumplan las condiciones especificadas en el antecedente, también deben cumplirse las condiciones especificadas en el consecuente [116]. Tanto antecedente como consecuente son conjunciones de átomos escritos de la forma $a_1 \wedge \cdots \wedge a_n$ [115]. Los átomos en estas reglas pueden tener la forma $C(x)$, $P(x, y)$, $sameAs(x, y)$ o $differentFrom(x, y)$, donde C es una clase OWL, P es una propiedad OWL y x , y son variables, individuos OWL o valores de datos OWL [115]. Por otro lado, las variables se definen utilizando un signo de interrogación como prefijo (por ejemplo,?x) [115].

Formalización de R1: Proceso en estado de Falla

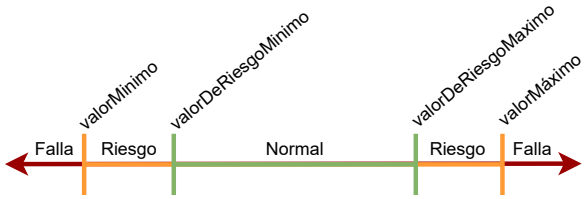

Figura 4.8: Rangos usados para la detección de anomalías

R1 está relacionado con estados de falla en los procesos. Como se describió en secciones previas, la detección de anomalías se realiza con base a las lecturas de cada variable, y los límites funcionales de esta última. Los requisitos de la regla consisten en obtener el valor censado en la clase Lectura por cada individuo de la clase Variable. Además, se debe extraer la información de los límites máximos y mínimos. Si las lecturas no están en los rangos óptimos de funcionamiento (Figura 4.8), se asigna un estado de falla al proceso correspondiente y el respectivo error diagnosticado. Los requisitos en R1 se formalizan en la Tabla 4.6

Regla	Nombre	SWRL
R11	Falla_Variable_Alto	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ tieneValorMaximo(?V, ?valmax) ∧ esCensadoPor(?V, ?S) ∧ tieneLectura(?S, ?L) ∧ tieneValorCensado(?L, ?val) ∧ swrlb:greaterThanOrEqual(?val, ?valmax) ∧ tieneLecturaAnomala(?E, ?L) ∧ tieneCategoria(?E, ?c) ∧ swrlb:equal(?c, “Alto”) → tieneEstado(?P, Falla) ∧ tieneEstado(?L, Alto) ∧ presentaError(?P, ?E)
R12	Falla_Variable_Bajo	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ tieneValorMinimo(?V, ?valmin) ∧ esCensadoPor(?V, ?S) ∧ tieneLectura(?S, ?L) ∧ tieneValorCensado(?L, ?val) ∧ swrlb:lessThanOrEqual(?val, ?valmin) ∧ tieneLecturaAnomala(?E, ?L) ∧ tieneCategoria(?E, ?c) ∧ swrlb:equal(?c, “Bajo”) → tieneEstado(?P, Falla) ∧ tieneEstado(?L, Bajo) ∧ presentaError(?P, ?E)
R13	Falla_Dilucion_Alto	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ VariableDeEntradaControlada(?V) ∧ tieneValorMaximo(?V, ?vmax) ∧ tieneValorPropuesto(?V, ?val) ∧ swrlb:greaterThanOrEqual(?val, ?vmax) ∧ afectaVariable(?E, ?V) → tieneEstado(?P, Falla) ∧ tieneEstado(?V, Alto) ∧ presentaError(?P, ?E)

R14	Falla_Dilucion_Bajo	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ VariableDeEntradaControlada(?V) ∧ tieneValorMinimo(?V, ?vmin) ∧ tieneValorPropuesto(?V, ?val) ∧ swrlb:lessThanOrEqual(?val, ?vmin) ∧ afectaVariable(?E, ?V) → tieneEstado(?P, Falla) ∧ tieneEstado(?V, Bajo) ∧ presentaError(?P, ?E)

Tabla 4.6: Reglas SWRL para estado de falla

Formalización de R2: Proceso en estado de Riesgo

R2 está definido para identificar si un proceso está en riesgo de presentar un error. Los requisitos de la regla consisten en obtener el valor censado en la clase Lectura por cada individuo de la clase Variable. Además, se debe extraer la información de los límites de riesgo máximo y mínimo. Si las lecturas no están en los rangos óptimos de riesgo (Figura 4.8), se asigna un estado de *Riesgo* al proceso correspondiente. Además, se define una relación entre el posible error y el proceso afectado a través de la propiedad de objeto *enRiesgoDePresentar*. Los requisitos en R2 se formalizan en la Tabla 4.7

Regla	Nombre	SWRL
R21	Riesgo_Variable_Alto	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ tieneValorRiesgoMaximo(?V, ?valrmax) ∧ tieneValorMaximo(?V, ?valmax) ∧ esCensadoPor(?V, ?S) ∧ tieneLectura(?S, ?L) ∧ tieneValorCensado(?L, ?val) ∧ swrlb:greaterThanOrEqual(?val, ?valrmax) ∧ swrlb:lessThan(?val, ?valmax) ∧ tieneLecturaAnomala(?E, ?L) ∧ tieneCategoria(?E, ?c) ∧ swrlb:equal(?c, "Alto") → tieneEstado(?P, Riesgo) ∧ tieneEstado(?L, Riesgo) ∧ enRiesgoDePresentar(?P, ?E)
R22	Riesgo_Variable_Bajo	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ tieneValorRiesgoMinimo(?V, ?valrmin) ∧ tieneValorMinimo(?V, ?valmin) ∧ esCensadoPor(?V, ?S) ∧ tieneLectura(?S, ?L) ∧ tieneValorCensado(?L, ?val) ∧ swrlb:greaterThan(?val, ?valmin) ∧ swrlb:lessThanOrEqual(?val, ?valrmin) ∧ tieneLecturaAnomala(?E, ?L) ∧ tieneCategoria(?E, ?c) ∧ swrlb:equal(?c, "Bajo") → tieneEstado(?P, Riesgo) ∧ tieneEstado(?L, Riesgo) ∧ enRiesgoDePresentar(?P, ?E)
R23	Riesgo_Dilucion_Alto	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ VariableDeEntradaControlada(?V) ∧ tieneValorRiesgoMaximo(?V, ?valrmax) ∧ tieneValorMaximo(?V, ?valmax) ∧ tieneValorPropuesto(?V, ?val) ∧ swrlb:greaterThanOrEqual(?val, ?valrmax) ∧ swrlb:lessThan(?val, ?valmax) ∧ afectaVariable(?E, ?V) → tieneEstado(?P, Riesgo) ∧ tieneEstado(?V, Riesgo) ∧ presentaError(?P, ?E)
R24	Riesgo_Dilucion_Bajo	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneVariable(?P, ?V) ∧ VariableDeEntradaControlada(?V) ∧ tieneValorRiesgoMaximo(?V, ?valrmax) ∧ tieneValorMaximo(?V, ?valmax) ∧ tieneValorPropuesto(?V, ?val) ∧ swrlb:greaterThanOrEqual(?val, ?valrmax) ∧ swrlb:lessThan(?val, ?valmax) ∧ afectaVariable(?E, ?V) → tieneEstado(?P, Riesgo) ∧ tieneEstado(?V, Riesgo) ∧ presentaError(?P, ?E)

Tabla 4.7: Reglas SWRL para estado de riesgo

Formalización de R3: Proceso en estado de Normal

R3 está definido para identificar si un proceso está teniendo un comportamiento normal con base a sus límites establecidos. Los requisitos de la regla consisten en obtener el valor censado en la clase Lectura por cada individuo de la clase Variable. Además, se debe extraer la información de los límites de riesgo máximo y mínimo. Si las lecturas están en los rangos óptimos de un funcionamiento normal (Figura 4.8), se asigna un estado *Normal* al proceso correspondiente. Los requisitos en R1 para el proceso DA se formalizan en la Tabla 4.8, para consultar el resto de las reglas SWRL consulte el Apéndice C

Regla	Nombre	SWRL
R31	Normal_DA	tieneValorRiesgoMaximo(Variable_Dil1_Entrada, ?vmax1) ∧ tieneValorRiesgoMinimo(Variable_Dil1_Entrada, ?vmin1) ∧ tieneValorPropuesto(Variable_Dil1_Entrada, ?val1) ∧ swrlb:greaterThan(?val1, ?vmin1) ∧ swrlb:lessThan(?val1, ?vmax1) ∧ tieneValorRiesgoMaximo(Variable_AGV_Entrada, ?vmax2) ∧ tieneValorRiesgoMinimo(Variable_AGV_Entrada, ?vmin2) ∧ tieneValorCensado(Lectura_AGV_Entrada, ?val2) ∧ swrlb:greaterThan(?val2, ?vmin2) ∧ swrlb:lessThan(?val2, ?vmax2) ∧ tieneValorRiesgoMaximo(Variable_DQO_Entrada, ?vmax3) ∧ tieneValorRiesgoMinimo(Variable_DQO_Entrada, ?vmin3) ∧ tieneValorCensado(Lectura_DQO_Entrada, ?val3) ∧ swrlb:greaterThan(?val3, ?vmin3) ∧ swrlb:lessThan(?val3, ?vmax3) ∧ tieneValorRiesgoMaximo(Variable_AGV_Salida, ?vmax4) ∧ tieneValorRiesgoMinimo(Variable_AGV_Salida, ?vmin4) ∧ tieneValorCensado(Lectura_AGV_Salida, ?val4) ∧ swrlb:greaterThan(?val4, ?vmin4) ∧ swrlb:lessThan(?val4, ?vmax4) ∧ tieneValorRiesgoMaximo(Variable_DQO_Salida, ?vmax5) ∧ tieneValorRiesgoMinimo(Variable_DQO_Salida, ?vmin5) ∧ tieneValorCensado(Lectura_DQO_Salida, ?val5) ∧ swrlb:greaterThan(?val5, ?vmin5) ∧ swrlb:lessThan(?val5, ?vmax5) ∧ tieneValorRiesgoMaximo(Variable_Biomasa_Salida, ?vmax6) ∧ tieneValorRiesgoMinimo(Variable_Biomasa_Salida, ?vmin6) ∧ tieneValorCensado(Lectura_Biomasa_Salida, ?val6) ∧ swrlb:greaterThan(?val6, ?vmin6) ∧ swrlb:lessThan(?val6, ?vmax6) → tieneEstado(Proceso_DA, Normal)

Tabla 4.8: Regla SWRL para estado normal en el proceso DA

4.5 Validación del modelo ontológico de detección de anomalías

Después de anexar las reglas SWRL en el modelo ontológico, nuestro modelo ahora es capaz de conectarse con un razonador y generar resultados de inferencia. Para validar tanto el modelo como las reglas SWRL seguimos un proceso en el cual primero alimentamos a individuos de la clase *Lectura*. Seguidamente, utilizando el razonador Pellet, ejecutamos las reglas SWRL. De esta forma, el razonador es capaz de inferir consecuencias lógicas de un conjunto de hechos o axiomas afirmados, y así, encontrar algún tipo de inconsciencia en la ontología.

Estado normal

La primera prueba realizada se hizo con lecturas de un estado normal, es decir, la lectura registrada de cada variable está dentro de los rangos (Tabla 2.4) óptimos de funcionamiento. La Tabla 4.9 presenta los valores usados por cada proceso.
Las Figuras 4.9 y 4.10 presentan una visualización en Proégé de los procesos DA y MEC posterior a la aplicación de las reglas SWRL usando Pellet. Podemos ver que las reglas infieren correctamente el estado normal de ambos procesos y no detecta ninguna

inconsistencia en la definición de la ontología. Además, a través de la relación inversa *esProcesoDe* de la propiedad de objeto *tieneProceso* se infieren a que biorrefinería pertenece cada proceso.

	DA									
No.	dil1	agv_in	dqo_in	biomasa	dqo_out	agv_out				
1	0,6	102.0	25.0	24.8	12.5	3945,50				
	MEC									
No.	agv_in	dil2	eapp	ace	xa	xm	xh	mox	imec	qh2
1	9890,54	1,16	0,65	6494,74	944,29	8,36E-291	1,18E-131	0,0057	0,0495	0,5136

Tabla 4.9: Medicion de un comportamiento normal para los proceso DA y MEC

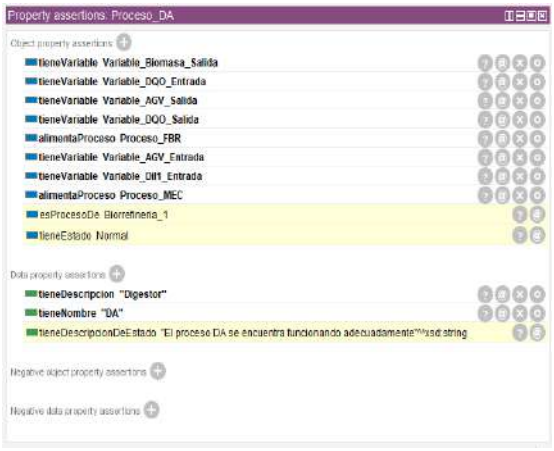

Figura 4.9: Protégé: Inferencia resultante pos-terior a la ejecución de reglas SWRL para el Proceso DA

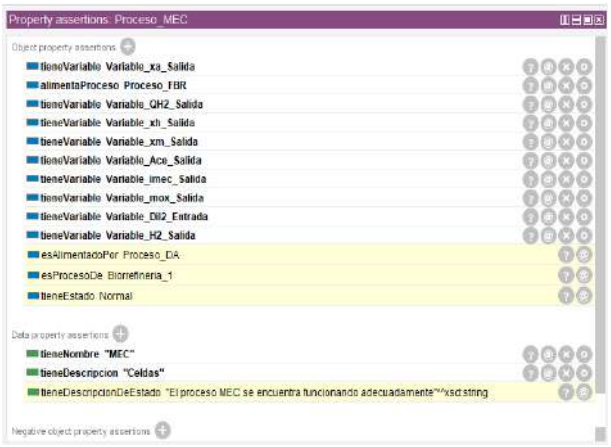

Figura 4.10: Protégé: Inferencia resultante poste-rior a la ejecición de reglas SWRL para el Proceso MEC

Mediciones mal clasificadas por RF

La siguiente prueba se realizó con mediciones que el clasificador RF descrito en la Sección 3.3 no pudo clasificar correctamente. De esta forma deseamos comparar la eficacia del modelo ontológico con respecto al modelo RF. Se compararon seis mediciones mal clasificadas por cada proceso. Las mediciones para el proceso DA se visualizan en la Tabla 4.10, mientras que para el proceso MEC en la Tabla 4.11
Como podemos apreciar, el modelo ontológico es capaz de clasificar correctamente aquellas mediciones que para el modelo RF son difíciles de predecir. Por ejemplo, en la Figura 4.11

	Variables						Estado		
No.	dil1	agv_in	dqo_in	biomasa	dqo_out	agv_out	RF	Ontología	True
1	0,966	65,49	27,65	6,63E-53	32,08	2945,50	0	1	1
2	0,475	69,11	25,16	2,77E-07	36,59	2938,22	0	1	1
3	0,314	75,37	25,28	1,94E-15	19,86	2991,05	0	1	1
4	0,646	101,61	30,01	2,23E-15	39,43	3001,00	1	0	0
5	0,713	106,48	80,00	284,38	25,40	16339,33	1	0	0
6	0,310	51,33	14,18	38,51	17,38	5237,49	1	0	0

Tabla 4.10: Mediciones registradas en el proceso DA

	Variables										Estado		
No.	agv_in	dil2	eapp	ace	xa	xm	xh	mox	imec	qh2	RF	Ontología	True
1	10110,2	3,36	0,59	6270,9	1008,01	1.9E-322	4,41E-277	0,0066	0,058	0,667	0	1	1
2	7760,9	3,09	0,54	4239,5	1002,98	2.4E-322	2,44E-153	0,0065	0,057	0,647	0	1	1
3	14750,6	3,54	0,59	10427,3	1022,11	2.08E-322	4,63E-201	0,0068	0,060	0,614	0	1	1
4	16699,8	1,22	0,16	11362,8	1396,20	2.03E-322	4,33E-225	0,0061	0,077	0,814	1	0	0
5	9897,4	2,37	0,44	4871,9	1374,54	2.08E-322	5,18E-174	0,0079	0,088	0,866	1	0	0
6	8954,8	1,69	0,27	3513,9	1319,91	9,79E-293	1,83E-111	0,0085	0,088	0,831	1	0	0

Tabla 4.11: Mediciones registradas en el proceso MEC

se presenta el resultado de inferencia para la medición no. 1 del proceso DA. Esta medición fue primeramente clasificada como un estado "normal" (0) por el modelo RF, sin embargo, la etiqueta real del estado es el de "anormal" (1). Por otro lado, el modelo ontológico fue capaz de determinar el estado verdadero dada la medición registrada. Además, a través de las relaciones establecidas, es posible extraer nueva información relevante al estado del proceso y realizar un diagnóstico del mismo. Por ejemplo, en este caso se ha determinado que el proceso DA presenta el error *Error_AGV_Salida_Bajo*, pero además está en riesgo de presentar otros dos errores. Por último, el modelo ontológico nos permite visualizar el alcance que puede llegar a tener un error hacia otros procesos. Por ejemplo, en la Figura 4.12 se aprecia como un proceso nuevo llamado FBR, que tiene una relación con el proceso DA, está en riesgo de ser afectado por un mal funcionamiento del proceso DA.
Este mismo resultado se puede observar en la Figura 4.13 para el proceso MEC, en este caso para la medición no. 2. De nuevo, el modelo ontológico es capaz de determinar el estado correcto dada la medición registrada. Por otro lado, en la Figura 4.14 vemos más información del error que se está presentando en el proceso MEC. La información incluye qué variables se están viendo afectadas, la recomendación a seguir para tratar de dar solución al error e información descriptiva del error.

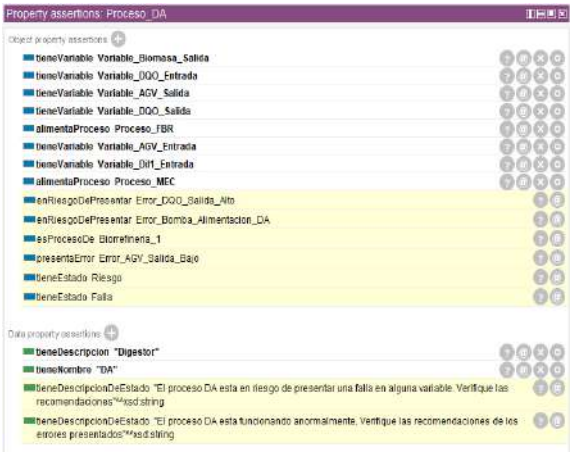

Figura 4.11: Protégé: Medición No. 1 del Pro-ceso DA

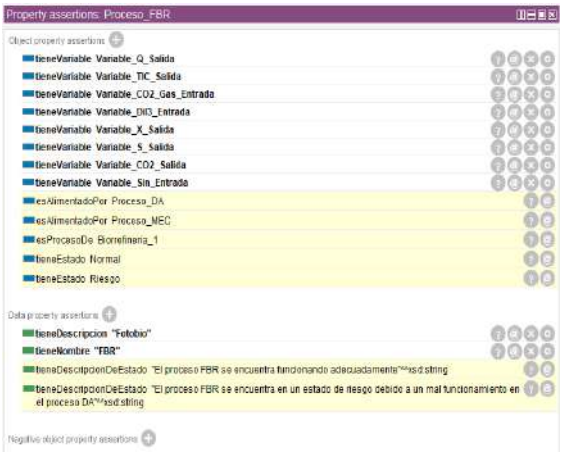

Figura 4.12: Protégé: Propagación del estado DA a otros procesos

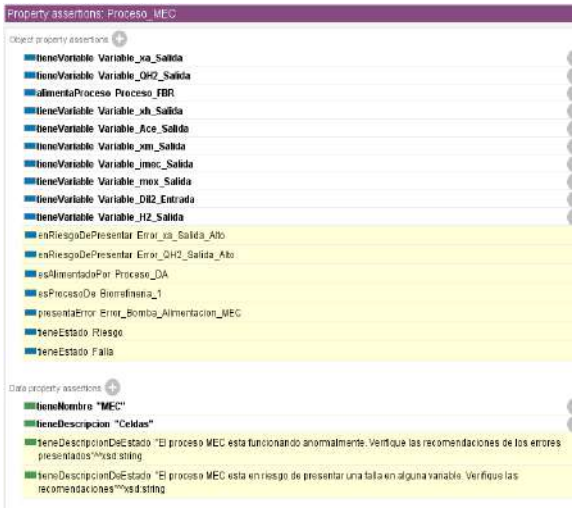

Figura 4.13: Protégé: Medición No. 2 del Pro-ceso MEC

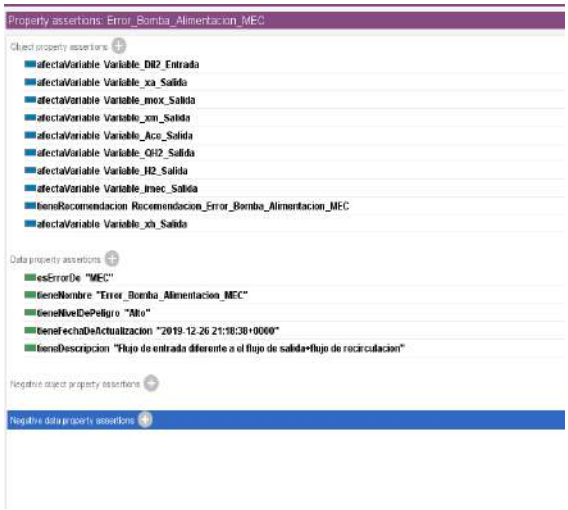

Figura 4.14: Protégé: - Error_Bomba_Alimentacion_MEC

Validación cruzada del modelo

Por último, se realizó una validación cruzada siguiendo el mismo proceso descrito en la Sección 3.3.5. Los resultados muestran que, si bien el clasificador RF tiene un porcentaje muy alto en las métricas de referencia usadas, el modelo ontológico supera estos puntajes (Tabla 4.12). Sin embargo, este comportamiento era esperado, dado que la metodología usada para la generación de errores se centró totalmente en los mínimos y máximos de cada variable. Por otro lado, el tiempo promedio que se toma el razonador Pellet en inferir

nueva información cada vez que una medición es agregada, es aproximadamente de 3.74 segundos (Figura 4.15). A pesar de que este tiempo no es excesivamente alto, si es un factor a considerar si la frecuencia en el registro de mediciones aumenta, pudiendo incluso provocar un desface entre el registro y la inferencia de los estados.

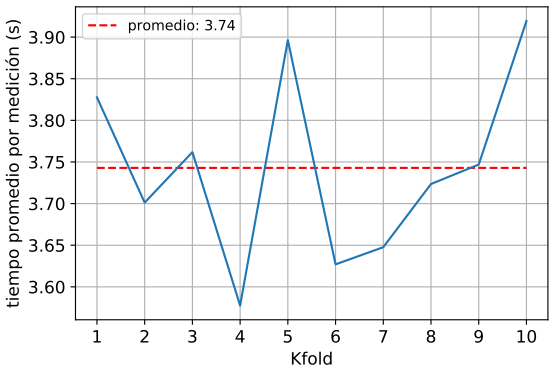

Figura 4.15: Tiempo de razonamiento por medición en cada *fold*

	DA										
Métrica	1-Fold	2-Fold	3-Fold	4-Fold	5-Fold	6-Fold	7-Fold	8-Fold	9-Fold	10-Fold	Promedio
accuracy	1	1	1	1	1	1	1	1	1	1	1
precision	1	1	1	1	1	1	1	1	1	1	1
recall	1	1	1	1	1	1	1	1	1	1	1
f1	1	1	1	1	1	1	1	1	1	1	1
	MEC										
accuracy	1	1	1	1	1	1	1	1	1	1	1
precision	1	1	1	1	1	1	1	1	1	1	1
recall	1	1	1	1	1	1	1	1	1	1	1
f1	1	1	1	1	1	1	1	1	1	1	1

Tabla 4.12: Puntajes obtenido usando k-fold cross-validation en el modelo ontológico

4.6 Modelo ontológico y modelos *Random Forest*

Hasta ahora hemos validado la eficacia de nuestro modelo ontológico en el escenario donde se cuenta con información de los límites de cada variable en un proceso. Sin embargo, en casos reales esta clase de información no suele estar disponible. En la sección 2.4.1 se describe una metodología para la obtención de estos límites. Sin embargo, no resulta práctico en una situación real donde no se asegura la existencia de un simulador del sistema complejo que nos interese. Por otro lado, los modelos RF descritos en el capítulo anterior, han sido entrenados usando los datos históricos de cada proceso. Por lo tanto, de forma implícita, los

modelos RF tienen conocimiento de los límites funcionales de cada variable. Sin embargo, extraer estos límites de forma directa requiere analizar cada regla de decisión creada por cada árbol en el bosque. Considerando que cada árbol es construido de forma diferente y que la profundidad puede llevarnos a bosques masivos de árboles, la interpretación se vuelve complicada (véase Figura 4.16).
Otra desventaja de los modelos RF en su estado original, es que solamente nos pueden decir el estado funcional de un proceso. No podemos saber que variable o grupo de variables pudieran estar involucradas en un estado anormal o de falla, ni el error que se estuviera presentando. Por otro lado, como se mostró en secciones previas, el modelo ontológico y las reglas de inferencia SWRL permiten inferir nuevas relaciones de los procesos en un estado de falla o riesgo.

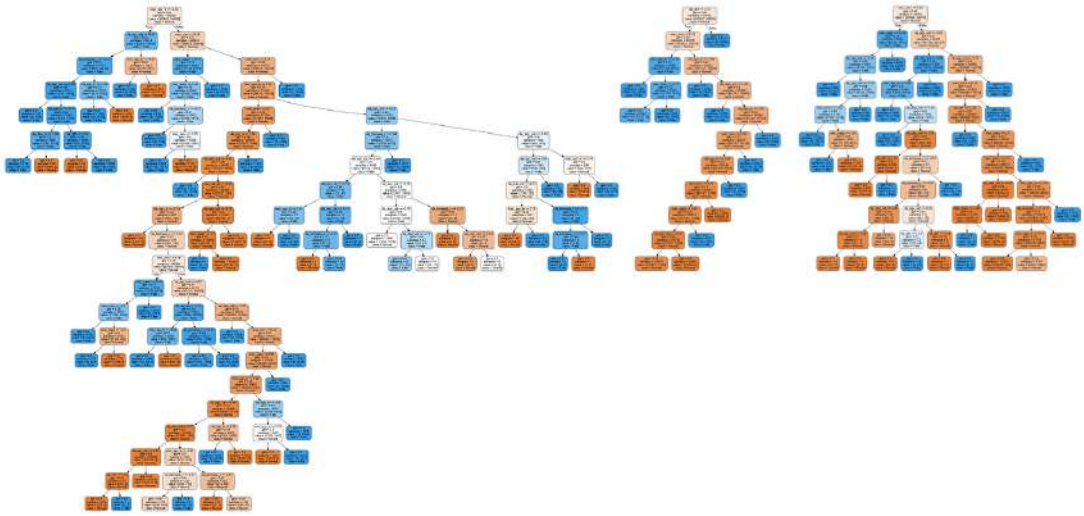

Figura 4.16: Bosque de árboles con tres estimadores (árboles de decisiones). A medida que el bosque crece, la interpretabilidad se vuelve compleja

Así, con las ventajas y desventajas de cada modelo, la solución que se plantea es hacer que la ontología realice el diagnóstico cuando los modelos RF detecten un estado inusual. Además, dado que el modelo ontológico requiere de conocer los límites de cada variable en un respectivo proceso, se propone hacer que este aprenda estos límites progresivamente a través de los modelos RF. Es decir, a medida que una medición es clasificada por los modelos RF, extraer los valores de cada variable y verificar si es un límite.

4.6.1 Integración y actualización de los límites

Sabemos que los modelos RF pueden determinar si un proceso funciona correctamente o de forma inusual con cierto nivel de fiabilidad. Un funcionamiento normal, se puede definir en función de las variables de un proceso. Si están en un rango de buen funcionamiento, el modelo RF lo clasifica como normal. Por lo contrario, si alguna variable o algún conjunto

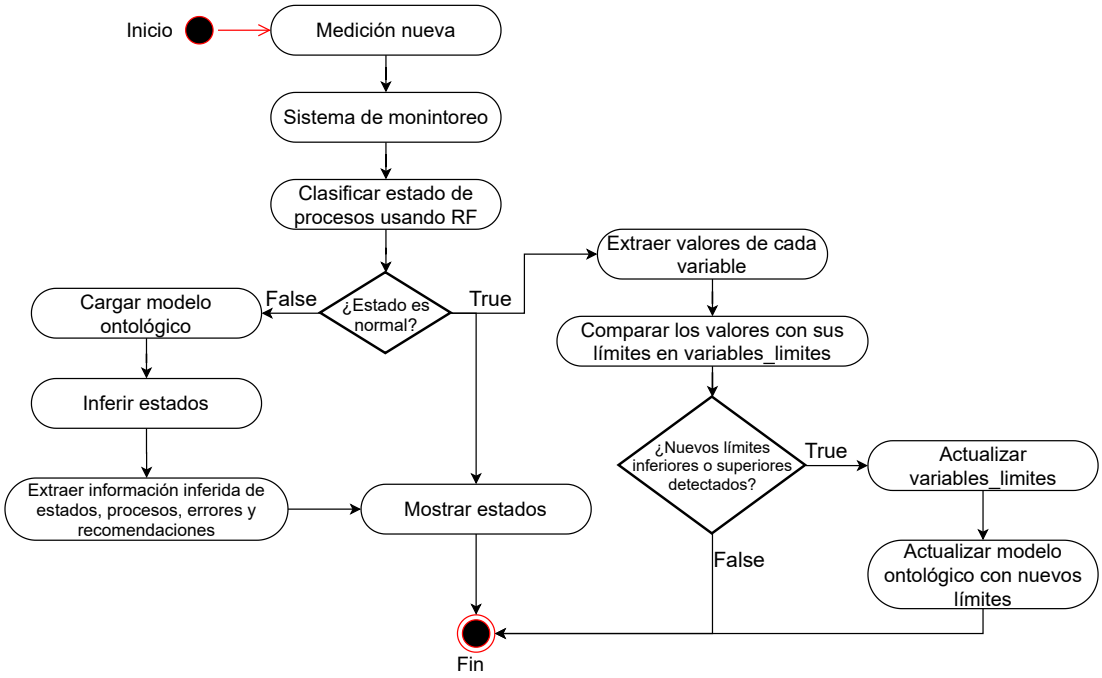

Figura 4.17: Secuencia de actualización de límites

de variables no están en los rangos óptimos, lo clasificaría como anormal.
Debido a que un estado anormal implicaría tener valores de variables fuera de los rangos de un funcionamiento normal, estos valores no funcionarían como punto de referencias para delimitar los rangos normales. Por otro lado, para los valores de variables de una medición normal se esperaría que estén dentro de un rango de funcionamiento aceptable. Así, estos nos pueden servir de guía para ir actualizando progresivamente los límites de las variables en el modelo ontológico. La actualización debería ocurrir, solo si el nuevo valor supera el límite previo almacenado. El esquema de secuencia básico de la integración de los modelos RF con el modelo ontológico y la actualización de los límites se describe en la Figura 4.17.

4.6.2 Resultados

En las Figuras 4.18 y 4.19 se presentan los resultados obtenidos de analizar 3000 mediciones siguiendo el esquema de la Figura 4.17. Podemos ver cómo, de forma progresiva y por cada medición nueva, las variables van actualizando sus límites desde un valor inicial hasta aproximarse al límite funcional implícito en los modelos RF. Si bien, no todas las variables se acercaron a los límites generados en la Sección 2.4.1, podemos ver, en la Tabla 4.13, que los límites aprendidos por el modelo ontológico son muy cercanos a los que podríamos considerar como los límites “ideales”.
Así, con ayuda de los modelos RF podemos lograr que nuestro modelo ontológico se adapte con el transcurso del tiempo al comportamiento dinámico de los procesos. Sin embargo, es importante mencionar que existe una alta dependencia de los modelos RF. Por lo

			Simulador		Aprendidos	
Variable	Unidad	Descripción	Máximo	Mínimo	Máximo	Mínimo
da_dil1	Ld^{-1}	Tasa de dilución	0.31	1.00	0.310018	0.999955
da_dqo_in	gL^{-1}	Dióxido de carbono	10.00	80.00	10.001153	79.995403
da_agv_in	$mmolL^{-1}$	Concentración de acetato	50.00	150.00	50.000156	149.993944
da_biomasa	gL^{-1}	Materia orgánica de origen vegetal o animal	0.00	415.18	0.0	325.655252
da_dqo_out	gL^{-1}	Dióxido de carbono	2.481	44.997	2.544701	44.989611
da_agv_out	$mmolL^{-1}$	Concentración de acetato	3002.60	26061.40	2944.076	20421.260
mec_agv_in	$mmolL^{-1}$	Concentración de acetato	3002.60	26061.40	2944.076	20421.260
mec_dil2	Ld^{-1}	Tasa de dilución	1.00	3.00	1.00001	3.46425
mec_eapp	V	Voltaje aplicado	0.10	0.60	0.100013	0.603606
mec_ace	$mmolL^{-1}$	Concentración de acetato dentro de la MEC	0.00	23284.90	228.546	18589.350
mec_xa	mgL^{-1}	Biomasa anodofílica	0.00	1396.26	331.504	1395.862
mec_xm	mgL^{-1}	Biomasa metanogénica	0.00	0.00017	8.39e-323	1.60e-24
mec_xh	mgL^{-1}	Biomasa hidrogenotrópica	0.00	14.36	1.23e-322	4.62e-07
mec_mox	L^{-1}	Mediador de oxidación	0.00	0.00889	0.000307	0.008894
mec_imec	A	Corriente	0.00	0.0881	0.001747	0.087761
mec_qh2	Ld^{-1}	Flujo de hidrógeno producido	0.00	0.86851	0.000802	0.86818

Tabla 4.13: Límites de funcionamiento óptimo por cada variable en los procesos DA y MEC

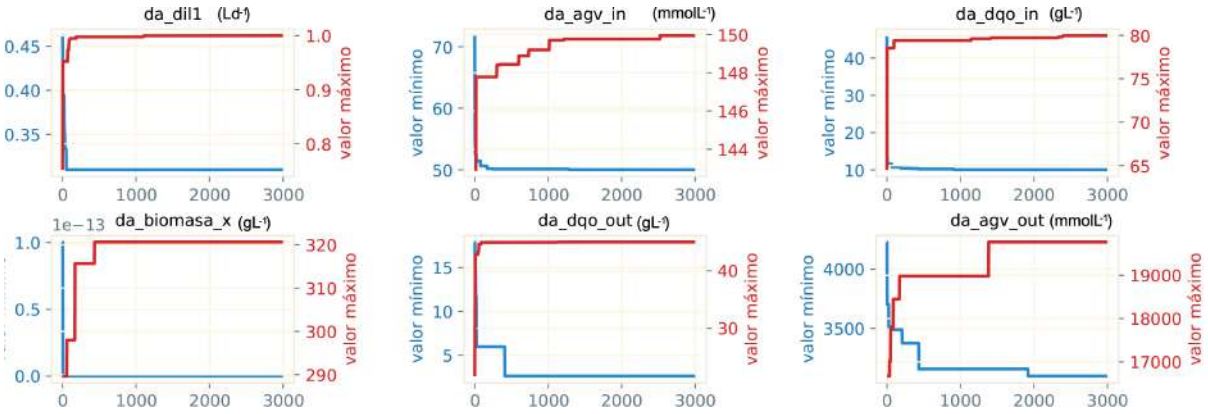

Figura 4.18: DA: Actualización de límites por cada proceso

tanto, se requiere que estos sean actualizados cada cierto tiempo para que el porcentaje de predicciones correctas se encuentre en niveles aceptables.

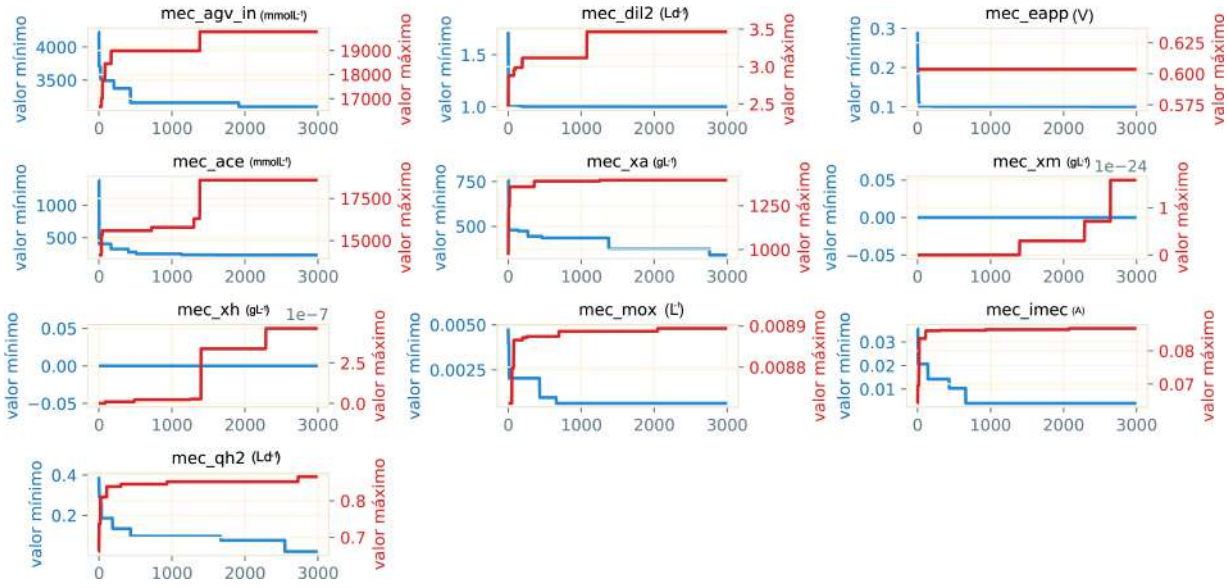

Figura 4.19: MEC: Actualización de límites por cada proceso

Capítulo 5

Sistema web de monitoreo y diagnóstico

Desde hace varios años la World Wide Web se ha convertido en un océano de datos e información y continúa creciendo sin cesar a un ritmo exponencial. Los sistemas y aplicaciones basados en la web ahora ofrecen una variedad compleja de contenido y funcionalidad variados a una gran cantidad de usuarios. Los servicios web proporcionan una infraestructura conocida e independiente de un idioma con el fin de integrar componentes heterogéneos. Sus estándares neutros, junto con tecnología de soporte, nos ayudan a enlazar componentes implementados usando diferentes lenguajes de programación [117]. Así, con los servicios web, los interesados pueden exponer sus procesos comerciales internos como servicios y hacerlos accesibles a través de Internet. Como ahora dependemos cada vez más de los sistemas y aplicaciones basados en la Web, su rendimiento, confiabilidad y calidad se han vuelto de una importancia primordial. Como resultado, el diseño, desarrollo, implementación y mantenimiento de sistemas basados en web se han vuelto más complejos y difíciles de administrar.
En este capítulo se describe la implementación del sistema web de monitoreo para la detección de anomalías y emisión de recomendaciones, de acuerdo con los modelos propuestos en capítulos anteriores. El objetivo principal del sistema web es el de otorgar a los operadores de la biorrefinería una herramienta para la asistencia en la toma de decisiones que sea fácil de utilizar, mantener y que sea escalable.

5.1 Caso de usos

El sitio web está pensado para otorgar las siguientes funcionalidades:

- Registrar mediciones.
- Determinar el estado actual e histórico de los procesos.
- Detectar errores en los procesos.

- Obtener recomendaciones para la toma de decisiones.
- Editar, agregar y eliminar errores y recomendaciones.
- Visualizar la información ontológica utilizada para el modelado del sistema de monitoreo.
- Predefinir los límites funcionales de las variables.
- Otorgar una API de la información de las mediciones para extender las funcionalidades básicas.

Para alcanzar estos objetivos, se propone el uso en conjunto de un modelo de aprendizaje automático y un modelo ontológico. Así, los casos de uso identificados se pueden resumir en la Figura 5.1.

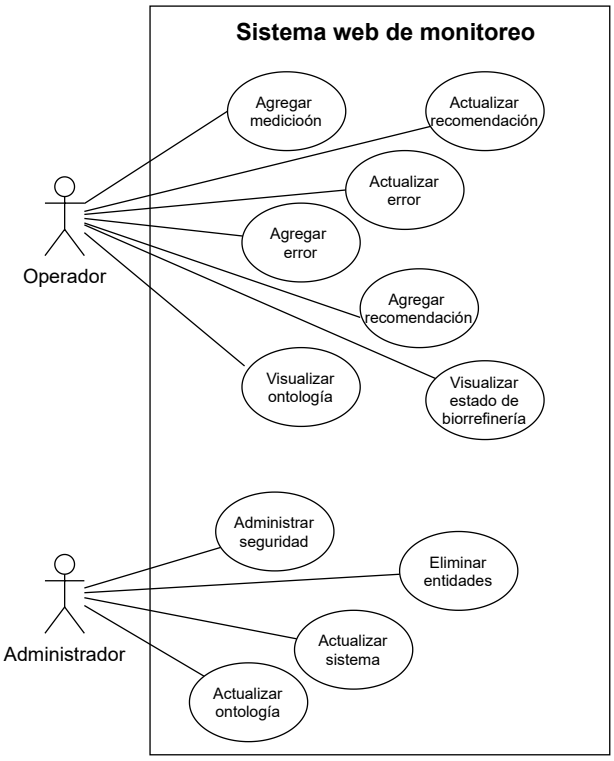

Figura 5.1: casos de uso

5.1.1 UC1: Agregar medición

Las mediciones son la fuente principal de datos para el funcionamiento del sistema de monitoreo y detección de anomalías. Debido a que se desconoce la estructura de la red de sensores en línea que se utilizará en la biorrefinería, se ha optado por suponer que el registro de mediciones se realizará de manera manual y cada 60 minutos. Por este motivo,

como actor principal tenemos a un operador, el cual debe ingresar a la plataforma web y solicitar el registro de una nueva medición. Por cada proceso activo, el operador debe registrar los valores medidos en cada variable. Posterior a que el operador finalice el registro, las mediciones serán almacenadas en el sistema e inmediatamente iniciará el proceso de análisis para determinar si el proceso está en un estado normal o anormal. Tras finalizar el análisis, el estado se le presentará al operador. Así, resumiendo los puntos importantes, tenemos que:

UC1: Agregar medición	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	La medición es guardado
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador inicia un registro nuevo de medición. 3. El operador ingresa el valor de cada variable por cada proceso activo. 4. El sistema guarda la medición y manda información al sistema de detección de anomalías. 5. El clasificador devuelve el estado general de la biorrefinería. Si es "normal" finaliza el proceso de registro, actualiza los límites de variables y muestra el estado. Si es "anormal" infiere los estados por proceso, los errores y recomendaciones usando reglas ontológicas.

Tabla 5.1: UC1

Diagrama de actividades (blackbox)

Uno de los objetivos principales del sistema es el de detectar anomalías en los procesos. Por ello se propone la utilización tanto del modelo Random Forest (RF) descrito en la Sección 3.3.4, como el uso del modelo ontológico descrito en la Sección 4.4. Ambos modelos siguiendo la estructura planteada en la Sección 4.6.
En la Figura 5.2 se describe superficialmente el proceso propuesto para lograr este fin. El primer paso, tras recibir el registro de una nueva medición, es limpiar los datos propios de la medición. Posteriormente, por cada proceso, aplicar los modelos RF. Si el estado es normal, se finaliza el registro y se presenta los resultados. Paralelamente, se ejecutará la actualización de los límites de cada variable para el modelo ontológico. Por otro lado, si el estado es anormal, se usará el modelo ontológico para realizar un diagnóstico más profundo de la anomalía. Para esto, usaremos las reglas SWRL descritas en el capítulo anterior y el razonador Pellet. Se extraerá la nueva información de los procesos y se presentarán los resultados al operador, finalizando el proceso de registro.

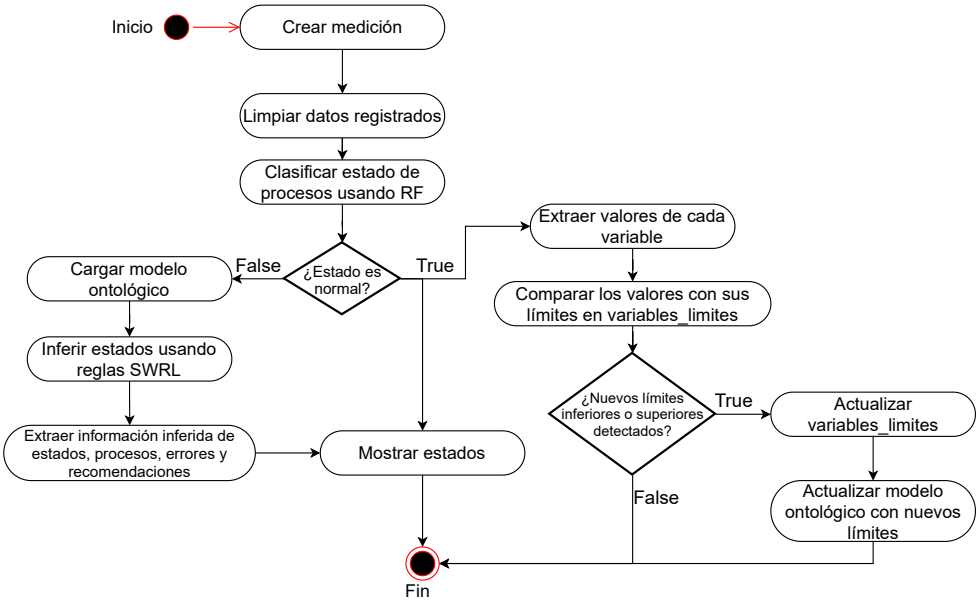

Figura 5.2: Diagrama de secuencia (blackbox): agregar medición

5.1.2 UC2: Visualizar el estado de la biorrefinería

Un punto importante para la toma de decisiones es el de poder visualizar toda la información relevante de los proceso de la biorrefinería. Por ejemplo, el estado de los procesos, mediciones históricas o errores presentes. Este proceso es iniciado con una solicitud hecha por un operador. El sistema debe responder de forma puntual con la información requerida.

UC2: Visualizar estado de biorrefinería	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	El estado de los procesos es mostrado correctamente
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador solicita información visual del panel de datos (dashboard). 3. El sistema recolecta información del estado de los procesos, errores y recomendaciones. 4. El sistema muestra la información solicitada.

Tabla 5.2: UC2

Diagrama de actividades (blackbox)

Para tomar decisiones el operador requiere de tener toda la información relevante a su alcance. La Figura 5.3 describe la idea propuesta. Primero, para facilitar la información se propone utilizar un panel de datos con gráficas y tablas de información. Para el acceso

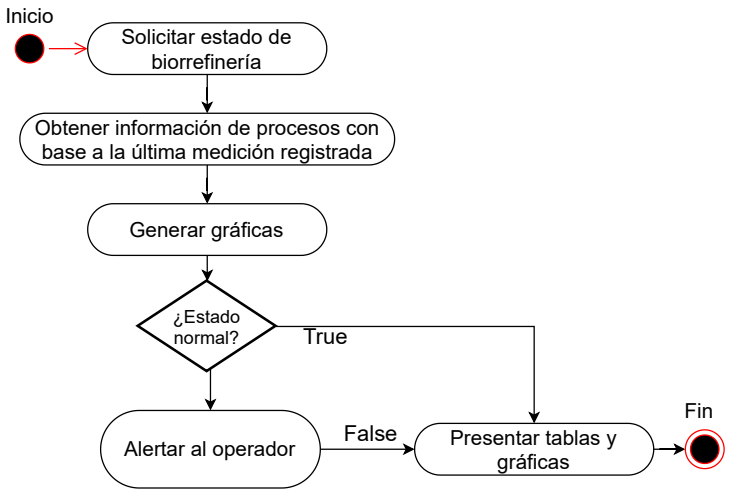

Figura 5.3: Diagrama de secuencia (blackbox): visualizar estado

a los datos se propone utilizar una interfaz API de las mediciones. Esta API retornará las mediciones en una estructura bien definida y óptimo para las gráficas. Se crea una gráfica por cada variable principal en cada uno de los procesos de la biorrefinería. Además, si el estado actual del proceso no es normal, se alertará al operador. En caso contrario, se procede a presentar las tablas y gráficas de los procesos con base a la última medición registrada.

5.1.3 UC3-6: Agregar y editar elementos de la ontología

El siguiente punto importante es el de poder agregar y editar algunos aspectos importantes del modelo ontológico. Por ejemplo, si el operador decidiese establecer fijo un límite inferior o superior de una variable, agregar nuevos errores o editar errores ya existentes. Debido a la importancia del modelo ontológico, el operador debe tener sumo cuidado con las modificaciones que se realice. Además, debido a la complejidad en sintaxis que conlleva la creación y edición de reglas, se considera conveniente no tener esta funcionalidad en las reglas SWRL. Al finalizar el proceso de agregación o edición de algún elemento ontológico, los cambios se almacenarán en el sistema web y, además, se realizarán los ajustes pertinentes en la ontología. En la Tabla D.2 se aprecia el UC para la agregación de errores. Para consultar el resto de los casos de uso, diríjase al Apéndice D

Diagrama de actividades (blackbox)

Otro objetivo deseado es el poder ofrecer un enlace entre el sistema de monitoreo y la ontología de la biorrefinería. De tal forma que, cada creación, actualización o eliminación realizada en el sistema de monitoreo de algún elemento ontológico, ya sea un error, recomendación o variable, éste se vea reflejado también en la ontología. Aunque en la Figura 5.4 se describe el proceso de actualizar un objeto, los procesos para agregar y eliminar

UC3: Agregar error	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	El error es guardado, la ontología es actualizada
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador comienza un nuevo registro de error. 3. El operador registra el nombre del error, las variables afectadas y una descripción del error. 4. El operador finaliza el registro del error. 5. El sistema guarda el error y envía información a la ontología. 6. La ontología se actualiza.

Tabla 5.3: UC3

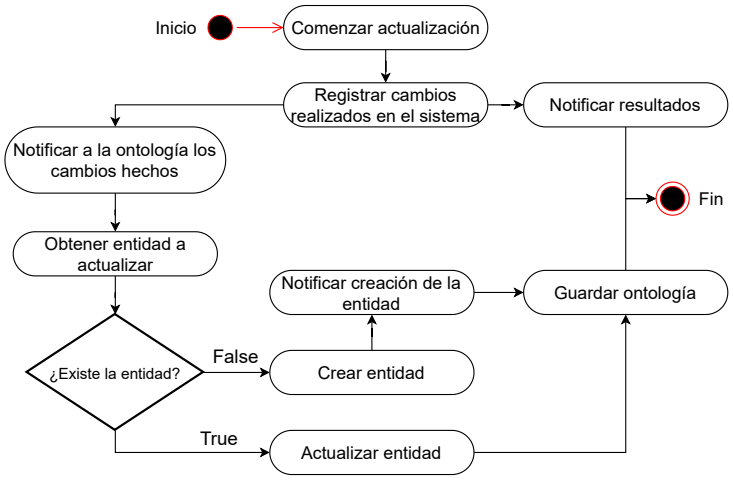

Figura 5.4: Diagrama de secuencia (blackbox): agregar y editar elementos ontológicos

siguen la misma estructura.

5.2 Implementación del sistema web

El sistema web ha sido creado bajo el framework de desarrollo web Django. Su arquitectura Model, View, Template, facilita el trabajo con los datos y la base de datos. La Figura 5.5 presenta la arquitectura de ejecución del sistema y los componentes que la conforman.

- **Sistema de monitoreo:** Aplicación Django usado como motor principal para la conexión entre el cliente y los recursos ofrecidos por el sistema. Entre sus funcionalidades está la de agregar mediciones y solicitar información histórica.
- **Detección y diagnóstico:** Otorga funcionalidades para determinar el estado general de la biorrefinería y sus procesos.

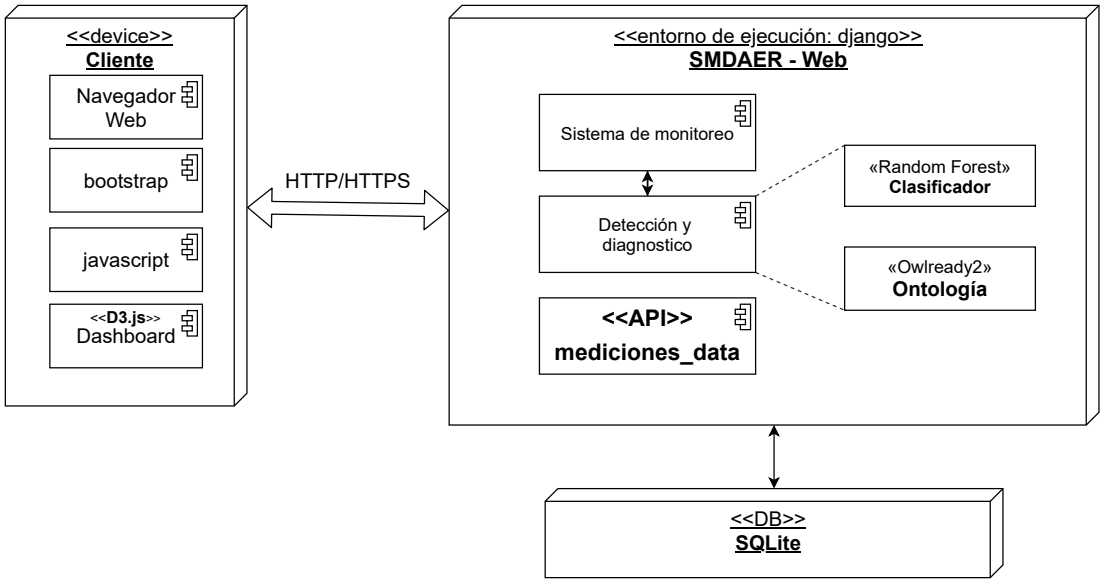

Figura 5.5: Arquitectura web para sistema de monitoreo y detección de fallas

- **API:** Proporciona la posibilidad de acceder a la información del sistema sin requerir el uso del mismo lenguaje de desarrollo. Esto permite el escalamiento en las funcionalidades del sistema al brindar datos de las mediciones almacenadas.

5.2.1 Componentes

Sistema de monitoreo

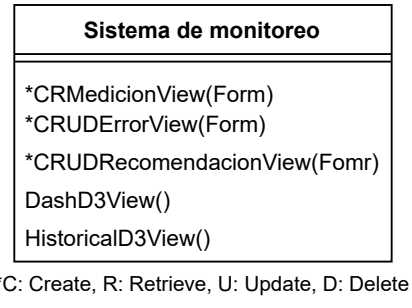

Figura 5.6: Sistema monitoreo

El sistema realiza la comunicación con el cliente usando el protocolo HTTP/HTTPS. El sistema lee las peticiones a través de una interfaz web (request) y devuelve el recurso solicitado (*View*). Este componente maneja la interacción entre la base de datos (*Model*), el modelo ontología y el modelo RF. Además, se encarga de la renderización de los recursos a través del uso de templates. La Figura 5.6 resume las funciones básicas detectadas en este componente.

Detección y diagnóstico

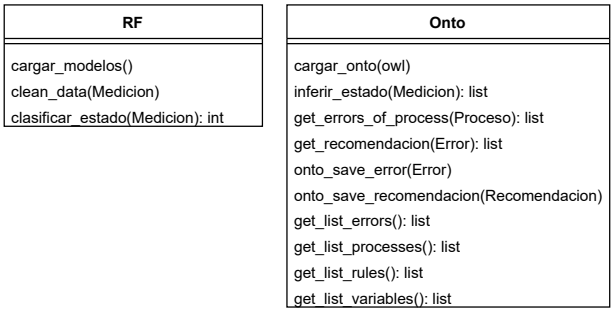

Figura 5.7: Detección y diagnóstico

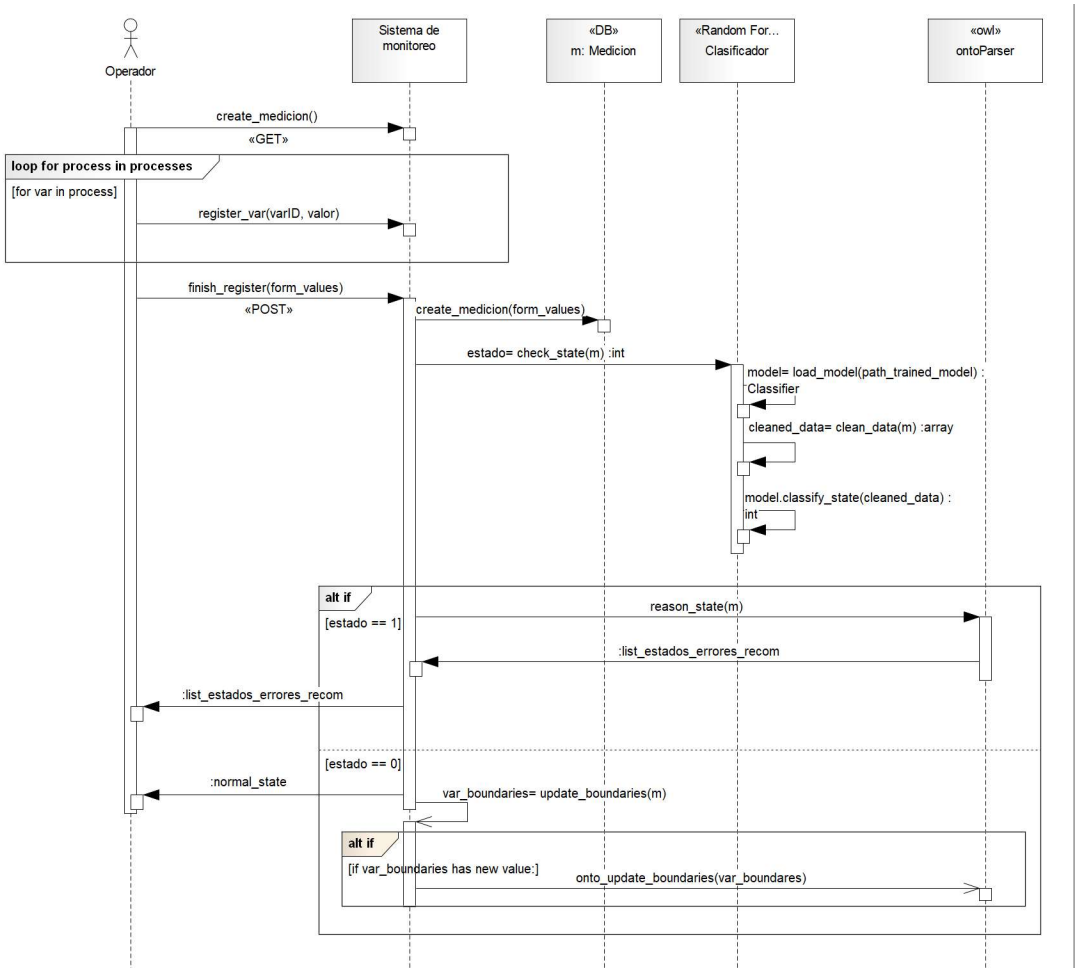

Figura 5.8: Diagrama de secuencia: agregar medición

Este módulo estará conformado por dos componentes. Los modelos RF y el modelo ontológico. Es invocado por el sistema de monitoreo cuando una medición es registrada. Además, el componente del modelo ontológico tendrá las herramientas para el manejo de la ontología a través del sistema web. Herramientas como, extracción de información de los procesos, transformarlo en estructuras de datos que facilitan su manejo, eliminación de entidades, etc. La Figura 5.7 presentan las funcionalidades principales de ambos componentes. Por otro lado, en la Figura E.3 podemos ver la interacción que se produce entre el sistema de monitoreo y el modulo de detección y diagnóstico. El proceso inicia con el operador enviando una solicitud de creación. El sistema valida los datos provenientes de un mensaje POST y crea un registro en la base de datos. Posteriormente, esta nueva medición pasa al Clasificador para obtener el estado de la biorrefinería. Si el estado es anormal o 1, inicia la interacción con la ontología a través del mensaje "reason_state" que devuelve una lista de datos estructurados de los procesos, errores y recomendaciones.

API

Lista de todas las mediciones	
URL	GET api/mediciones/ HTTP/1.1
Accept	application/json

Agregar medición	
URL	POST api/medicion/ HTTP/1.1
Accept	application/json
Body	• date:date • da_dil1:float • da_agv_in:float • da_dqo_in:float • da_biomasa_x:float • da_dqo_out:float • da_agv_out:float • mec_agv_in:float • mec_dil2:float • mec_eapp:float • mec_ace:float • mec_xa:float • mec_xm:float • mec_xh:float • mec_mox:float • mec_imec:float • mec_qh2:float

Obtener medición	
URL	GET api/medicion/id HTTP/1.1
Accept	application/json

Tabla 5.4: Recursos disponibles a través de la API

Una API permite que otras personas dentro o fuera de una organización pueda hacer uso de los servicios o productos que se ofrecen para crear nuevas aplicaciones o expandir la aplicación principal. Las API internas mejoran la productividad de los equipos de desarrollo al maximizar la reutilización y aplicar la coherencia en nuevas aplicaciones. Para propósitos de este proyecto, se implementó una interfaz API que facilita el acceso a los datos de las mediciones. De esta forma, lo que se busca es poder facilitar la adición de nuevos módulos sin tener que ingresar directamente al código principal. Por ejemplo, para este proyecto se ha utilizado un panel de datos que es alimentado puramente por los datos de la API. Las funcionalidades básicas de la API se pueden ver en la Tabla 5.4.

5.2.2 Recursos del sistema

Con los requisitos descritos previamente, se han desarrollado una serie de recursos que el sistema web brindará a los operadores. Un recurso lo consideramos como una vista en el entorno de trabajo Django. La vista se encarga de procesar los datos y generar templates para presentar la información. Por ejemplo, el path / está asociado a la vista D3DashView. Esta vista se encarga de recolectar todos los datos para generar la información necesaria para generar un dashboard y presentar al operador el estado de la biorrefinería. La lista completa se puede ver en la Tabla 5.5. En *Path*, se describe la URL para acceder al recurso, en *Recurso* encontramos a que vista interna en Django está enlazada el path, por último una breve descripción del objetivo del recurso.

5.3 Pruebas y resultados

5.3.1 Sistema web de monitoreo y detección

Primero, se ha probado el sistema web con una medición que representa un estado normal. En la Figura 5.9 se observa la vista obtenida posterior al registro de la medición. En el recuadro #1 encontramos los valores de la medición registrada. En el recuadro #2, está el estado de la biorrefinería, así también el estado de los procesos que fueron inferidos por el clasificador RF y el modelo ontológico. En el recuadro #3 se presenta, mediante un grafo, los estados de cada proceso. El estado es indicado mediante el color; por ejemplo, verde para un estado normal, naranja para uno de riesgo y rojo para un estado de falla. Además, el grafo es interactivo, es decir, el operador puede ir expandiendo cada nodo para obtener más información del proceso. Por último, en el recuadro #4 se presenta un historial de mediciones pasadas en un par de tablas. Cada tabla tiene la capacidad de filtrar las mediciones usando como parámetro de búsqueda, cualquiera de las columnas representadas. La Figura 5.10 se presenta el caso en el que una anomalía se ha detectado. En el recuadro #2 se aprecia el diagnóstico obtenido del modelo ontológico y las reglas SWRL. El nivel de peligro de cada error es representado por un color, donde el color rojo representa el nivel más alto de peligro y el azul el nivel más bajo. En el recuadro #3 se presenta la información del recuadro #2 de forma gráfica. Podemos ver más profundidad en el grafo a comparación del estado normal de la Figura 5.9.

Path	Recurso(smdaer_app)	Descripción
/	.views.D3DashView	Página principal (Dashboard)
/<str:task_id>	.views.D3DashView	Página principal (nuevo registro)
/admin/	django.contrib.admin.sites.index	página para administrador
/api/mediciones/	.api.views.MedicionListAPIView	API:Solicitar todas las mediciones
/api/medicion/	.api.views.MedicionCreateAPIView	API:Crear medición
/api/medicion/<int:pk>	.api.views.MedicionRetrieveAPIView	API:Solicitar una medición
/api/recomendacion/<int:epk>/	.api.views.RecomendacionDetailAPIView	API:Obtener un recomendación dado un error
/api/tree_data/<int:mpk>/	.api.views.TreeDataAPIView	API:Obtener datos para el grafo de estados
/error/	.views.ErrorListView	Lista de todos los errores
/error/<int:pk>/	.views.ErrorDetailView	Detalles de un error
/error/<int:pk>/delete/	.views.ErrorDeleteView	Eliminación de un error
/error/<int:pk>/update/	.views.ErrorUpdateView	Actualización de un error
/error/<int:pk>/update/<str:action>	.views.ErrorAssignRecomendacionView	Actualizar una recomendación a través de la creación de un error
/error/create/	.views.ErrorCreateView	Crear un error
/get-task-info/	.views.get_task_info	Obtener actualización de un task asíncrono
/historical/	.views.DCHistoricalDataView	Información de mediciones anteriores
/medicion/<int:pk>/	.views.MedicionDetailView	Detalle de una medición
/medicion/create/	.views.MedicionCreateView	Crear una medición
/ontologia/	.views.ontology	Información del modelo ontológico
/proceso/	.views.ProcesoListView	:proceso-list
/recomendacion/<int:pk>/delete/	.views.RecomendacionDeleteView	Eliminación de una recomendación
/recomendacion/<int:pk>/update/<int:epk>/	.views.RecomendacionUpdateView	Actualización de una recomendación
/recomendacion/<int:rpk>/unassign/<int:epk>/	.views.unassign_recomendacion	Desasignar una recomendación de un error
/recomendacion/create/<int:epk>/	.views.RecomendacionCreateView	Crear una recomendación desde la vista de error
/regla/	.views.ReglaListView	Lista de reglas
/regla/<int:pk>/	.views.ReglaDetailView	Detalle de una regla

Tabla 5.5: Recursos disponibles en el sistema web

Así, con los resultados obtenidos podemos ver que el sistema web cumple con las funcionalidades deseadas de detección y diagnóstico.

5.3.2 API

Previamente definimos que el API desarrollado fue pensado para escalar el sistema web y poder dar facilidades para la adición de nuevos módulos a partir de los datos de las mediciones. Con esta idea, se presentan a continuación el resultado de utilizar dos de los recursos definidos. Primero, en la Figura 5.11 se presenta el resultado obtenido de utilizar el recurso POST /api/medicion/ para el registro de una medición sin el uso de la interfaz web. Por otro lado, en la Figura 5.12 se observa la respuesta del servidor al solicitar el recurso GET /api/medicion/1044, el cual nos devuelve la información de la medición registrada previamente.

5.3.3 Interacción con el modelo ontológico

Otro objetivo deseado para el sistema web es el de evitar que el operador tenga que entrar directamente al código fuente para realizar algún cambio con respecto al sistema

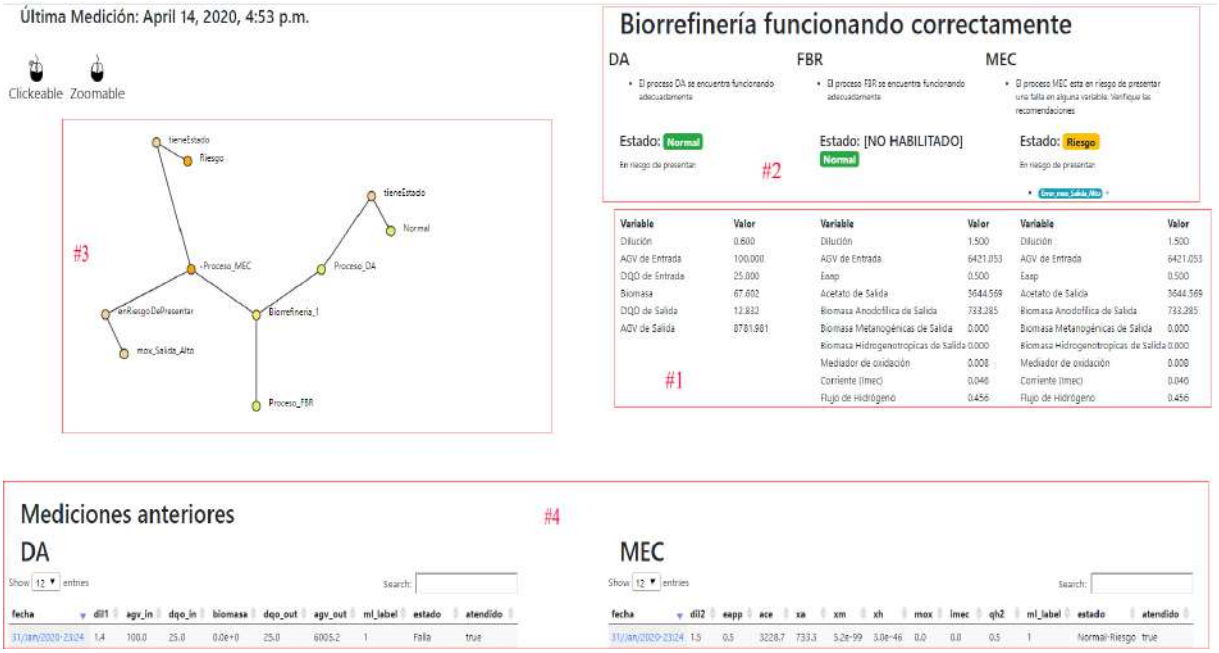

Figura 5.9: Sistema Web - Estado normal

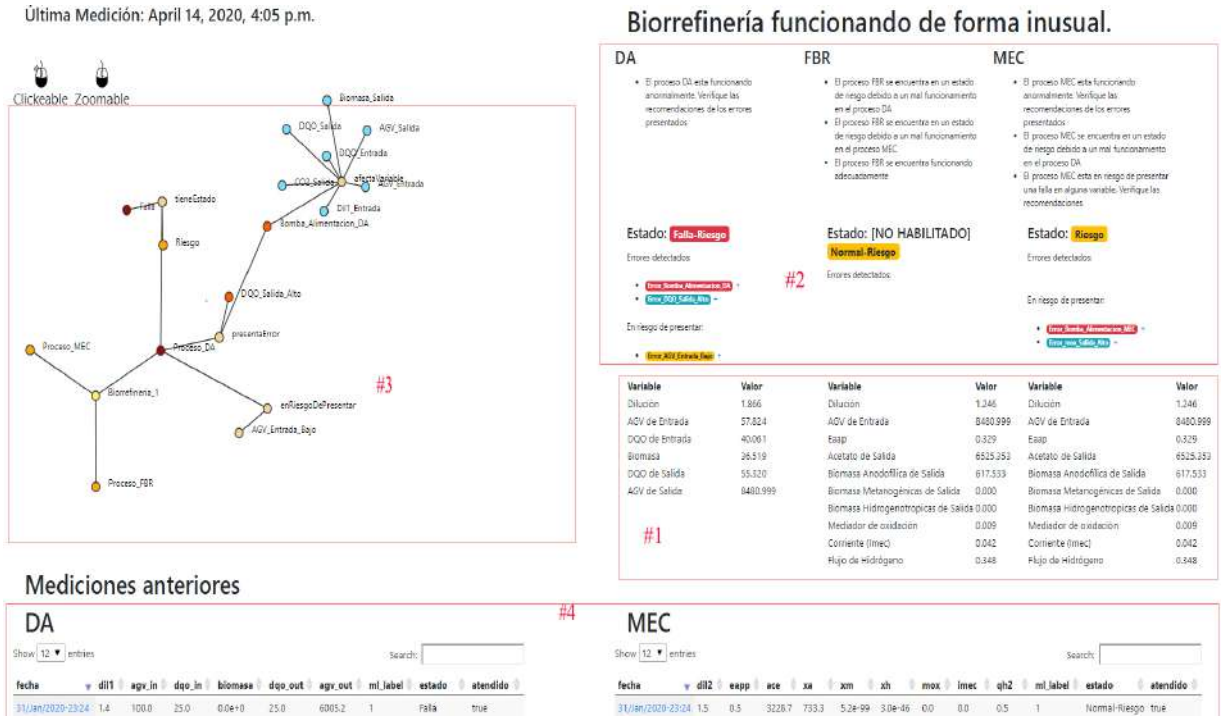

Figura 5.10: Sistema Web - Estado anormal

Figura 5.11: Recurso API para crear una medición

Figura 5.12: Recurso API para consultar una medición

de detección y diagnóstico guiado por el modelo ontológico. Para ello, se implementó la funcionalidad de realizar cambios directos al modelo a través del sistema web. De esta forma, el operador no requiere tener conocimiento alguno del lenguaje ontológico para realizar cambios o actualizaciones del modelo. Para ejemplificar esta prueba, se creó un error en el sistema web (Figura 5.13) y se utilizó Protégé para verificar la actualización del modelo ontológico (Figura5.14).

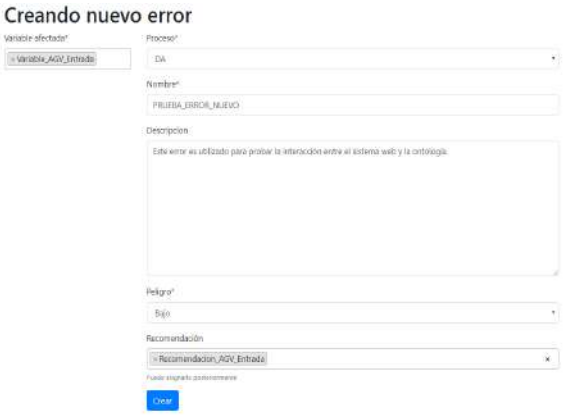

Figura 5.13: Creación de un error en el sis-tema web

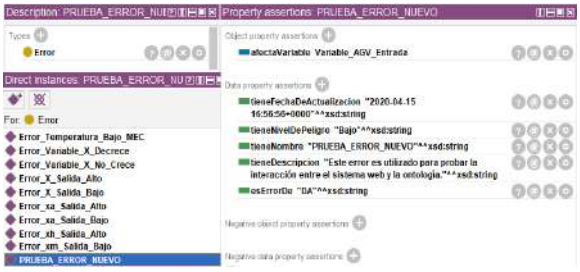

Figura 5.14: Creación de un error en el mo-delo ontológico a través del sistema web

En el siguiente fragmento de consola, podemos ver la interacción que hubo entre el sistema web y el modelo ontológico. Por ejemplo, vemos en la línea 2 que se ha mandado a llamar al método *create_error_async* con los parámetros de la línea 3. El método se encarga de convertir estos parámetros a un formato adecuado para el modelo para posteriormente crear el individuo *PRUEBA_ERROR_NUEVO* . Dado que, en la creación del error asignamos una recomendación, se procede a llamar al método *onto_update_recomendacion* (línea 8) para agregar el nuevo error a la lista del individuo *Recomendacion_AGV_Entrada* (línea 9-32).

```
1  [ 11:56:57,516: ] Received task: sistema_monitoreo_app.tasks.run_ontology_create_error_async[588e801b-2
       fee-4f17-ba38-a33786a70560]
2  [ 11:56:57,870: ] inside create_error_async
3  [ 11:56:58,106: ]{'nombre': 'PRUEBA_ERROR_NUEVO', 'descripcion': 'Este error es utilizado para probar la
       interacción entre el sistema web y la ontología.', 'peligro': 'Bajo', 'es_error_de': 'DA', '
       variables': ['Variable_AGV_Entrada'], 'c
4  reated_at': '2020-04-15 16:56:56+0000'}
5  [ 11:56:58,115: ]Adding bio_abril_10_2020_10am.Variable_AGV_Entrada to bio_abril_10_2020_10am.
       PRUEBA_ERROR_NUEVO
6  [ 11:56:58,255: ]Ontology: Error bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO created
7  [ 11:56:58,264: ]Received task: sistema_monitoreo_app.tasks.run_ontology_update_recomendacion_async[
       ba9d5d78-3b55-47c4-945a-cb68f4abb400]
8  [ 11:56:58,354: ]Inside ONTO_UPDATE_RECOMENDACION
9  [ 11:56:58,354: ]recomendacion: {'nombre': 'Recomendacion_AGV_Entrada', 'descripcion': 'Verificar la
       concentracion de AGV en la entrada.', 'iri': 'http://www.uady.mx/1052668570/ontologias/2019/8/
       biorrefineria#Recomendacion_AGV_Entrada',
10 'errores': ['Error_AGV_Entrada_Alto', 'Error_AGV_Entrada_Bajo', 'PRUEBA_ERROR_NUEVO'], 'updated_at': '
       2020-04-15 16:56:57+0000'}
11 [ 11:56:58,355: ]old_name: None
12 [ 11:56:58,356: ]fields_updated: error_from_createview
13 [ 11:56:58,358: ]Error_form_request: [bio_abril_10_2020_10am.Error_AGV_Entrada_Alto,
       bio_abril_10_2020_10am.Error_AGV_Entrada_Bajo, bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO]
14 [ 11:56:58,359: ]List cleared
15 [ 11:56:58,359: ]New elements added
16 [ 11:56:58,360: ]err: Error_AGV_Entrada_Alto
17 [ 11:56:58,360: ]Adding error Error_AGV_Entrada_Alto to bio_abril_10_2020_10am.Recomendacion_AGV_Entrada
18 [ 11:56:58,360: ]onto_error_name: bio_abril_10_2020_10am.Error_AGV_Entrada_Alto
19 [ 11:56:58,361: ]esRecomendacionDe: [bio_abril_10_2020_10am.Error_AGV_Entrada_Alto,
       bio_abril_10_2020_10am.Error_AGV_Entrada_Bajo, bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO]
20 [ 11:56:58,361: ]Error_AGV_Entrada_Alto is already added
21 [ 11:56:58,362: ]err: Error_AGV_Entrada_Bajo
22 [ 11:56:58,362: ]Adding error Error_AGV_Entrada_Bajo to bio_abril_10_2020_10am.Recomendacion_AGV_Entrada
23 [ 11:56:58,362: ]onto_error_name: bio_abril_10_2020_10am.Error_AGV_Entrada_Bajo
24 [ 11:56:58,363: ]esRecomendacionDe: [bio_abril_10_2020_10am.Error_AGV_Entrada_Alto,
       bio_abril_10_2020_10am.Error_AGV_Entrada_Bajo, bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO]
25 [ 11:56:58,364: ]Error_AGV_Entrada_Bajo is already added
26 [ 11:56:58,364: ]err: PRUEBA_ERROR_NUEVO
27 [ 11:56:58,365: ]Adding error PRUEBA_ERROR_NUEVO to bio_abril_10_2020_10am.Recomendacion_AGV_Entrada
28 [ 11:56:58,365: ]onto_error_name: bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO
29 [ 11:56:58,366: ]esRecomendacionDe: [bio_abril_10_2020_10am.Error_AGV_Entrada_Alto,
       bio_abril_10_2020_10am.Error_AGV_Entrada_Bajo, bio_abril_10_2020_10am.PRUEBA_ERROR_NUEVO]
30 [ 11:56:58,366: ]PRUEBA_ERROR_NUEVO is already added
31 [ 11:56:58,366: ]New elements added from ErrorCreateView
32 [ 11:56:58,419: ]Ontology: recomendacion bio_abril_10_2020_10am.Recomendacion_AGV_Entrada updated
33 [ 11:56:58,459: ]Task sistema_monitoreo_app.tasks.run_ontology_create_error_async[588e801b-2fee-4f17-
       ba38-a33786a70560] succeeded in 0.92200000000204891s: None
34 [ 11:56:58,465: ]Task sistema_monitoreo_app.tasks.run_ontology_update_recomendacion_async[ba9d5d78-3b55
       -47c4-945a-cb68f4abb400] succeeded in 0.203000000000861473s: None
```

5.4 Discusiones sobre resultados

Con las pruebas realizadas, podemos decir que el sistema web cumple con los requisitos básicos planteados en un principio. Brinda al operador las funciones de agregar mediciones, visualizar estados y facilitar la actualización de individuos en el modelo ontológico. Además, posibilita el escalamiento con la API interna. Sin embargo, el tiempo de respuesta agregado por la inferencia del modelo ontológico, ralentiza la presentación de la información en al

menos 3.74 segundos posterior a la creación de una medición nueva. Si se considera que la recepción de nuevas mediciones es cada hora, esto no representa un problema. Pero, si el intervalo de recepción de mediciones disminuye a un nivel menor a este tiempo, el sistema web podría presentar problemas para manejar las peticiones.

Capítulo 6

Conclusiones

La complejidad biológica asociada a los procesos de una biorrefinería resulta en sistemas dinámicos descritos por modelos altamente no lineales de orden mayor a dos. Además, los procesos que los constituyen suelen operar con un gran número de variables y debido a su naturaleza altamente correlacionada, la presencia de una anomalía en un proceso específico puede influir en otros a través de la propagación de la misma.
Monitorear el estado de los procesos y detectar cualquier falla se considera un punto clave en la correcta operación de cualquier proceso industrial y en particular de una biorrefinería. Aunque las fallas se pueden analizar casi en tiempo real utilizando los datos recopilados y modelos de aprendizaje automático, esto es sólo la mitad del problema. El segundo, igualmente importante, es diagnosticar el error, su alcance e interpretarlo de tal forma que un operador puede hacer uso de la información. Para este punto, las ontologías han mostrado ser una herramienta importante para razonar las relaciones existentes (y nuevas) entre las entidades del proceso modelado.
En este trabajo de tesis hemos presentado la propuesta de una metodología para la creación de un sistema de detección de anomalías y emisión de recomendaciones en una biorrefinería de producción de hidrógeno. Nuestro enfoque propone la utilización de un modelo de aprendizaje automático (*Random Forest*) para detectar fallas y un modelo ontológico para realizar la interpretación y el diagnostico.
Para alcanzar este objetivo, primero se implementó una versión más rápida del simulador utilizado para la obtención de datos de una biorrefinería con dos procesos acoplados. Nuestra versión mostró ser 11 veces más rápido que la versión original y tener nuevas características, por ejemplo, la inyección de ruido en las variables. Entre las ventajas destaca el poder realizar más experimentos en menos tiempo y la facilidad de escalar la complejidad del modelo.
Posteriormente, con los datos generados por el simulador, se utilizó *Random Forest* (RF) para la detección de estados normales y anormales. RF tiene la ventaja de que funciona notablemente bien con muy poca información, y con una baja probabilidad a sobre entrenarse, por lo tanto, es adecuado para datos altamente automatizados.
Los resultados experimentales mostraron que *Random Forest* clasifica correctamente estados normales y anormales en cada proceso. Esto se sustenta con puntajes altos de *precisión*,

sensibilidad y *puntaje F1.*
Por otra parte, sabemos que un proceso biológico esta formado de un gran número de variables de estado y que estas variables se relacionan con otras más a través de las reacciones bioquímicas involucradas. El conocer que un proceso es normal o anormal resulta poco informativo y sin mucha utilidad. Detectar el origen de la falla y su alcance es una tarea desafiante utilizando solo *Random Forest*, debido a los inmensos árboles de decisión que se pueden llegar a generar. Por ello, se propone el uso de un modelo ontológico conformado por clases, entidades reales (individuos) e interacciones entre clases (relaciones). Estos elementos agregan un significado y facilitan la adquisición automática de nueva información que es necesaria para encontrar el origen y alcance de una falla.
El modelo propuesto convierte el conocimiento de cada proceso en la biorrefinería en estructuras semánticas utilizada para diversas tareas, como la recuperación automática de información e inferencias de estados. Este modelo se centra en los procesos de la biorrefinería propuesta, Digestor Anaerobío y Celdas Electrolíticas Microbianas. De igual forma, se modelan las relaciones de los componentes de cada proceso, por ejemplo, la relación entre variables y sensores. Además, se proponen una serie de reglas SWRL para la inferencia de nueva información. Así, con el uso del razonador Pellet, el modelo puede detectar una falla y extraer un diagnóstico sobre la información de los errores o posibles fallas en cada proceso.
Las pruebas experimentales mostraron que el modelo ontológico es capaz de detectar anomalías que el modelo *Random Forest* no fue capaz de clasificar correctamente. Además, dada la expresividad de la ontología, podemos extraer nueva información de un estado anormal. Por ejemplo, qué variable se comporta de forma anormal, qué variables se están viendo afectadas, el nivel de peligro que representa esta anomalía o qué recomendaciones se deben seguir para la resolución del problema. Esto otorga una capa extra de interpretabilidad al modelo RF. Así, la combinación de ambos modelos permite satisfacer el flujo básico de un sistema de detección de fallas (detectar, evaluar, diagnosticar y recuperar).
La desventaja básica de este enfoque es la dependencia de las reglas SWRL con los límites de las variables involucradas en los procesos. Esta información no suele estar disponible de manera directa, pero si implícita en los datos. Nuestro enfoque propone utilizar los modelos RF como medio de aprendizaje para la obtención de estos límites por parte del modelo ontológico. Los resultados mostraron que a medida que las mediciones entren al sistema, el modelo ontológico va adaptando sus límites hasta aproximarse a los ideales.
Como resultado final, se presenta un sistema web de monitoreo y detección de anomalías que utiliza tanto el modelo RF, como modelo el ontológico. Este sistema web otorga funcionalidades para mejorar el proceso de toma de decisiones por parte de los operadores de la biorrefinería. Funcionalidades tales como:

- Registrar mediciones.
- Determinar el estado actual de los procesos.
- Detectar errores en los procesos.
- Obtener recomendaciones para la toma de decisiones.

- Facilitar la interacción entre un operador y el modelo ontológico.

Además, el sistema propuesto tiene la capacidad de ampliarse a través del uso de la API implementada. Esto permite añadir nuevas funcionalidades a partir de las mediciones registradas por los operadores. Por ejemplo, algún módulo estadístico o la utilización directa de los modelos para la detección y diagnóstico.
Así, es posible afirmar que todos los objetivos planteados en este proyecto de tesis se cumplieron con el desarrollo de la plataforma web, dado que se consiguió obtener un sistema funcional que permita monitorear los procesos de la biorrefinería, detectar anomalías y emitir recomendaciones.
Nuestro enfoque no propone el reemplazo total de un operador en el proceso de diagnóstico de fallas en los procesos, sino el incentivar la utilización en conjunto de un modelo de aprendizaje automático y un modelo ontológico, como apoyo para el operador en la toma de decisiones con el fin de mejorar el proceso de diagnóstico.
Por otra parte, a pesar de que los resultados obtenidos en este trabajo de tesis son satisfactorios, cabe mencionar que los datos usados provienen de un simulador y no reflejan con detalle la naturaleza compleja de una biorrefinería. Esto se ve reflejado en los puntajes altos obtenidos por los modelos. Por lo tanto, como trabajo a futuro se sugiere experimentar con los siguientes enfoques: 1) mejorar la precision del sistema con información real del funcionamiento de la biorrefinería, 2) ampliar la base del conocimiento del modelo ontológico para considerar nuevos niveles de complejidad en los procesos, 3) implementar un esquema de re entrenamiento para los modelos de predicción y mejorar el desempeño del sistema web, 4) mejorar el acoplamiento del simulador utilizando los tres procesos de la biorrefinería propuestos originalmente.

Apéndice A

Simulador con inyección aleatoria de ruido

Inyección de ruido aleatorio durante la ejecución de la simulación. En la Figura A.1 se observa el flujo de simulación con la inyección de ruido agregada. Esta segunda inyección genera ruido mayor a dos desviaciones estándar, lo que puede llegar a inducir a que las variables sobrepasen sus límites establecidos anteriormente y producirse valores anómalos.

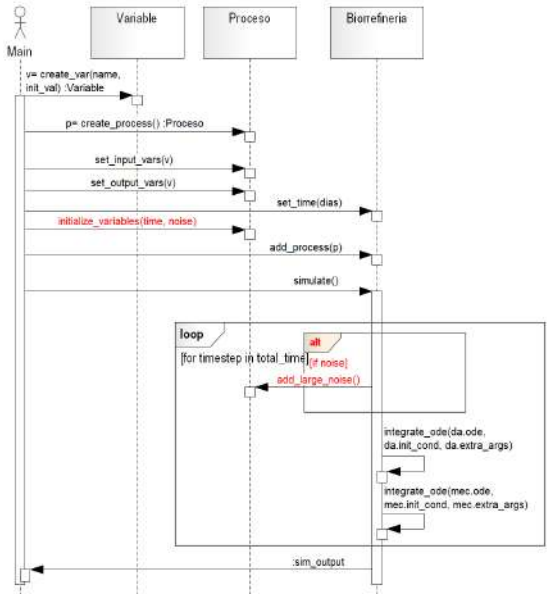

Figura A.1: Generador de errores induciendo variables de entrada a estados anormales

Apéndice B

Descripción de clases en lenguaje DL

$$Biorrefineria \equiv \top \sqcap$$
$$(\exists\ tieneProceso.Proceso) \sqcap$$
$$(= 1\ tieneNombre.string) \sqcap$$
$$(Sensor \sqsubseteq \bot) \sqcap$$
$$(Error \sqsubseteq \bot) \sqcap$$
$$(Proceso \sqsubseteq \bot) \sqcap$$
$$(Estado \sqsubseteq \bot) \sqcap$$
$$(Variable \sqsubseteq \bot) \sqcap$$
$$(Recomendacion \sqsubseteq \bot) \sqcap$$
$$(Lectura \sqsubseteq \bot)$$

$$Proceso \equiv \top \sqcap$$
$$(\exists\ tieneEstado.Estado) \sqcap$$
$$(\exists\ tieneVariable.Variable) \sqcap$$
$$(\forall\ alimentaProceso.Proceso) \sqcap$$
$$(\forall\ tieneError.Error) \sqcap$$
$$(= 1\ tieneNombre.string) \sqcap$$
$$(= 1\ tieneDescripcion.string) \sqcap$$
$$(= 1\ tieneDescripcionDeEstado.string) \sqcap$$
$$(= 1\ tieneFechaDeActualizacion.string) \sqcap$$
$$(Sensor \sqsubseteq \bot) \sqcap$$
$$(Error \sqsubseteq \bot) \sqcap$$
$$(Biorrefineria \sqsubseteq \bot) \sqcap$$
$$(Estado \sqsubseteq \bot) \sqcap$$
$$(Variable \sqsubseteq \bot) \sqcap$$
$$(Recomendacion \sqsubseteq \bot) \sqcap$$
$$(Lectura \sqsubseteq \bot)$$

$$Variable \equiv \top \sqcap$$
$$(= 1\ esCensadoPor.Sensor) \sqcap$$
$$(= 1\ tieneNombre.string) \sqcap$$
$$(= 1\ tieneDescripcion.string) \sqcap$$
$$(= 1\ tieneFechaDeActualizacion.string) \sqcap$$
$$(= 1\ tieneUnidadDeMedida.string) \sqcap$$
$$(= 1\ tieneValorMinimo.float) \sqcap$$
$$(= 1\ tieneValorMaximo.float) \sqcap$$
$$(= 1\ tieneValorRiesgoMinimo.float) \sqcap$$
$$(= 1\ tieneValorRiesgoMaximo.float) \sqcap$$
$$(Sensor \sqsubseteq \bot) \sqcap$$
$$(Error \sqsubseteq \bot) \sqcap$$
$$(Biorrefineria \sqsubseteq \bot) \sqcap$$
$$(Proceso \sqsubseteq \bot) \sqcap$$
$$(Estado \sqsubseteq \bot) \sqcap$$
$$(Recomendacion \sqsubseteq \bot) \sqcap$$
$$(Lectura \sqsubseteq \bot)$$

$$
\begin{aligned}
Error \equiv \top \sqcap \\
(\exists\ esErrorDe.Proceso) \sqcap \\
(\exists\ afectaVariable.Variable) \sqcap \\
(\exists\ tieneLecturaAnomala.Lectura) \sqcap \\
(\exists\ tieneRecomendacion.Recomendacion) \sqcap \\
(= 1\ tieneNombre.string) \sqcap \\
(= 1\ tieneDescripcionDeError.string) \sqcap \\
(= 1\ tieneFechaDeActualizacion.string) \sqcap \\
(= 1\ tieneNivelDePeligro.string) \sqcap \\
(Variable \sqsubseteq \bot) \sqcap \\
(Lectura \sqsubseteq \bot) \sqcap \\
(Biorrefineria \sqsubseteq \bot) \sqcap \\
(Proceso \sqsubseteq \bot) \sqcap \\
(Estado \sqsubseteq \bot) \sqcap \\
(Recomendacion \sqsubseteq \bot) \sqcap \\
(Sensor \sqsubseteq \bot)
\end{aligned}
$$

$$
\begin{aligned}
Recomendacion \equiv \top \sqcap \\
(\exists\ esRecomendacionDe.Error) \sqcap \\
(= 1\ tieneNombre.string) \sqcap \\
(= 1\ tieneDescripcionDeRecomendacion.string) \sqcap \\
(= 1\ tieneFechaDeActualizacion.string) \sqcap \\
(Variable \sqsubseteq \bot) \sqcap \\
(Lectura \sqsubseteq \bot) \sqcap \\
(Biorrefineria \sqsubseteq \bot) \sqcap \\
(Proceso \sqsubseteq \bot) \sqcap \\
(Estado \sqsubseteq \bot) \sqcap \\
(Error \sqsubseteq \bot) \sqcap \\
(Sensor \sqsubseteq \bot)
\end{aligned}
$$

Apéndice C

SWRL

Regla	Nombre	SWRL
R32	Normal_MEC	tieneValorRiesgoMaximo(Variable_xh_Salida, ?vmax1) ∧ tieneValorRiesgoMinimo(Variable_xh_Salida, ?vmin1) ∧ tieneValorCensado(Lectura_xh_Salida, ?val1) ∧ swrlb:greaterThan(?val1, ?vmin1) ∧ swrlb:lessThan(?val1, ?vmax1) ∧ tieneValorRiesgoMaximo(Variable_QH2_Salida, ?vmax2) ∧ tieneValorRiesgoMinimo(Variable_QH2_Salida, ?vmin2) ∧ tieneValorCensado(Lectura_QH2_Salida, ?val2) ∧ swrlb:greaterThan(?val2, ?vmin2) ∧ swrlb:lessThan(?val2, ?vmax2) ∧ tieneValorRiesgoMaximo(Variable_xm_Salida, ?vmax3) ∧ tieneValorRiesgoMinimo(Variable_xm_Salida, ?vmin3) ∧ tieneValorCensado(Lectura_xm_Salida, ?val3) ∧ swrlb:greaterThan(?val3, ?vmin3) ∧ swrlb:lessThan(?val3, ?vmax3) ∧ tieneValorRiesgoMaximo(Variable_mox_Salida, ?vmax4) ∧ tieneValorRiesgoMinimo(Variable_mox_Salida, ?vmin4) ∧ tieneValorCensado(Lectura_mox_Salida, ?val4) ∧ swrlb:greaterThan(?val4, ?vmin4) ∧ swrlb:lessThan(?val4, ?vmax4) ∧ tieneValorRiesgoMaximo(Variable_H2_Salida, ?vmax5) ∧ tieneValorRiesgoMinimo(Variable_H2_Salida, ?vmin5) ∧ tieneValorCensado(Lectura_H2_Salida, ?val5) ∧ swrlb:greaterThan(?val5, ?vmin5) ∧ swrlb:lessThan(?val5, ?vmax5) ∧ tieneValorRiesgoMaximo(Variable_Ace_Salida, ?vmax6) ∧ tieneValorRiesgoMinimo(Variable_Ace_Salida, ?vmin6) ∧ tieneValorCensado(Lectura_Ace_Salida, ?val6) ∧ swrlb:greaterThan(?val6, ?vmin6) ∧ swrlb:lessThan(?val6, ?vmax6) ∧ tieneValorRiesgoMaximo(Variable_xa_Salida, ?vmax7) ∧ tieneValorRiesgoMinimo(Variable_xa_Salida, ?vmin7) ∧ tieneValorCensado(Lectura_xa_Salida, ?val7) ∧ swrlb:greaterThan(?val7, ?vmin7) ∧ swrlb:lessThan(?val7, ?vmax7) ∧ tieneValorRiesgoMaximo(Variable_imec_Salida, ?vmax8) ∧ tieneValorRiesgoMinimo(Variable_imec_Salida, ?vmin8) ∧ tieneValorCensado(Lectura_imec_Salida, ?val8) ∧ swrlb:greaterThan(?val8, ?vmin8) ∧ swrlb:lessThan(?val8, ?vmax8) ∧ tieneValorRiesgoMaximo(Variable_Dil2_Entrada, ?vmax9) ∧ tieneValorRiesgoMinimo(Variable_Dil2_Entrada, ?vmin9) ∧ tieneValorPropuesto(Variable_Dil2_Entrada, ?val9) ∧ swrlb:greaterThan(?val9, ?vmin9) ∧ swrlb:lessThan(?val9, ?vmax9) → tieneEstado(Proceso_MEC, Normal)

Tabla C.1: Reglas SWRL para estado normal - MEC

Regla	Nombre	SWRL
R4	Descripcion_Estado_Falla	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneNombre(?P, ?n1) ∧ swrlb:stringConcat(?inicio, .El proceso ", ?n1) ∧ swrlb:stringConcat(?finDescripcion, ?inicio, .esta funcionando anormalmente. Verifique las recomendaciones de los errores presentados") ∧ tieneEstado(?P, Falla) → tieneDescripcionDeEstado(?P, ?finDescripcion)

Tabla C.2: Regla SWRL utilizada para generar un texto que describa el estado de un proceso cuando está fallando

Regla	Nombre	SWRL
R5	Descripcion_Estado_Normal	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneNombre(?P, ?n1) ∧ swrlb:stringConcat(?inicio, .El proceso ", ?n1) ∧ swrlb:stringConcat(?finDescripcion, ?inicio, "se encuentra funcionando adecuadamente") ∧ tieneEstado(?P, Normal) → tieneDescripcionDeEstado(?P, ?finDescripcion)

Tabla C.3: Regla SWRL utilizada para generar un texto que describa el estado de un proceso cuando está normal

Regla	Nombre	SWRL
R6	Descripcion_Estado_Riesgo	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ tieneNombre(?P, ?n1) ∧ enRiesgoDePresentar(?P, ?E) ∧ swrlb:stringConcat(?inicio, .El proceso ", ?n1) ∧ swrlb:stringConcat(?finDescripcion, ?inicio, .esta en riesgo de presentar una falla en alguna variable. Verifique las recomendaciones") ∧ tieneEstado(?P, Riesgo) → tieneDescripcionDeEstado(?P, ?finDescripcion)

Tabla C.4: Regla SWRL utilizada para generar un texto que describa el estado de un proceso cuando está en riesgo

Regla	Nombre	SWRL
R7	Descripcion_Riesgo_alimentaProceso	Biorrefineria(?B) ∧ tieneProceso(?B, ?P) ∧ alimentaProceso(?P, ?P2) ∧ tieneNombre(?P, ?n1) ∧ tieneNombre(?P2, ?n2) ∧ swrlb:stringConcat(?inicio, .El proceso ", ?n2) ∧ swrlb:stringConcat(?verboSujeto, ?inicio, "se encuentra en un estado de riesgo") ∧ swrlb:stringConcat(?frase, ?verboSujeto, "debido a un mal funcionamiento en el proceso ") ∧ swrlb:stringConcat(?finDescripcion, ?frase, ?n1) ∧ tieneEstado(?P, Falla) → tieneEstado(?P2, Riesgo) ∧ tieneDescripcionDeEstado(?P2, ?finDescripcion)

Tabla C.5: Regla SWRL utilizada para generar un texto que describa el estado de un proceso cuando está en riesgo a través de la propagación del error

Regla	Nombre	SWRL
R8	Todos_Procesos_Umbral_Peligro	Variable(?var) ∧ tieneValorMaximo(?var, ?vmax) ∧ swrlb:multiply(?x, ?vmax, 0.7) ∧ tieneValorMinimo(?var, ?vmin) ∧ swrlb:multiply(?y, ?vmin, 1.3) → tieneValorRiesgoMaximo(?var, ?x) ∧ tieneValorRiesgoMinimo(?var, ?y)

Tabla C.6: Regla SWRL utilizada para generar el rango los limites de riesgo mínimo y riesgo máximo

Apéndice D

Casos de uso

UC1: Agregar medición	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	La medición es guardado
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador inicia un registro nuevo de medición. 3. El operador ingresa el valor de cada variable por cada proceso activo. 4. El sistema guarda la medición y manda información al sistema de detección de anomalías. 5. El clasificador devuelve el estado general de la biorrefinería. Si es "normal" finaliza el proceso de registro y muestra el estado. Si es "anormal" infiere los estados por proceso, los errores y recomendaciones usando reglas ontológicas.

Tabla D.1: UC1

UC2: Visualizar estado de biorrefinería	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	El estado de los procesos es mostrado correctamente
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador solicita información visual del panel de datos (dashboard). 3. El sistema recolecta información del estado de los procesos, errores y recomendaciones. 4. El sistema muestra la información solicitada.

Tabla D.2: UC2

UC3: Agregar error	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	El error es guardado, la ontología es actualizada
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador comienza un nuevo registro de error. 3. El operador registra el nombre del error, las variables afectadas y una descripción del error. 4. El operador finaliza el registro del error. 5. El sistema guarda el error y envía información a la ontología. 6. La ontología se actualiza.

Tabla D.3: UC3

UC4: Agregar recomendación	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	La recomendación es guardada, la ontología es actualizada
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador comienza un nuevo registro de recomendación. 3. El operador registra el nombre del error, el o los errores asociados a la recomendación y una descripción de la recomendación. 4. El operador finaliza el registro de la recomendación. 5. El sistema guarda el recomendación y envía información a la ontología. 6. La ontología se actualiza.

Tabla D.4: UC4

UC5: Actualizar error o recomendación	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	El error o recomendación es guardada, la ontología es actualizada.
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador ingresa a la sección correspondiente para actualizar un error o recomendación. 3. El operador realiza un cambio en el elemento seleccionado. 4. El operador finaliza la actualización. 5. El sistema guarda la actualización y envía información a la ontología. 6. La ontología se actualiza.

Tabla D.5: UC5

UC6: Visualizar ontología	
Actor principal	Operador
Pre-condiciones	El operador es identificado y autenticado
Garantía de éxito	La ontología es mostrada sin error.
Escenario principal	1. El operador ingresa al sistema de monitoreo. 2. El operador solicita información de la ontología. 3. El sistema recupera información de la ontología y la muestra.

Tabla D.6: UC6

Apéndice E

Diagrama de secuencia

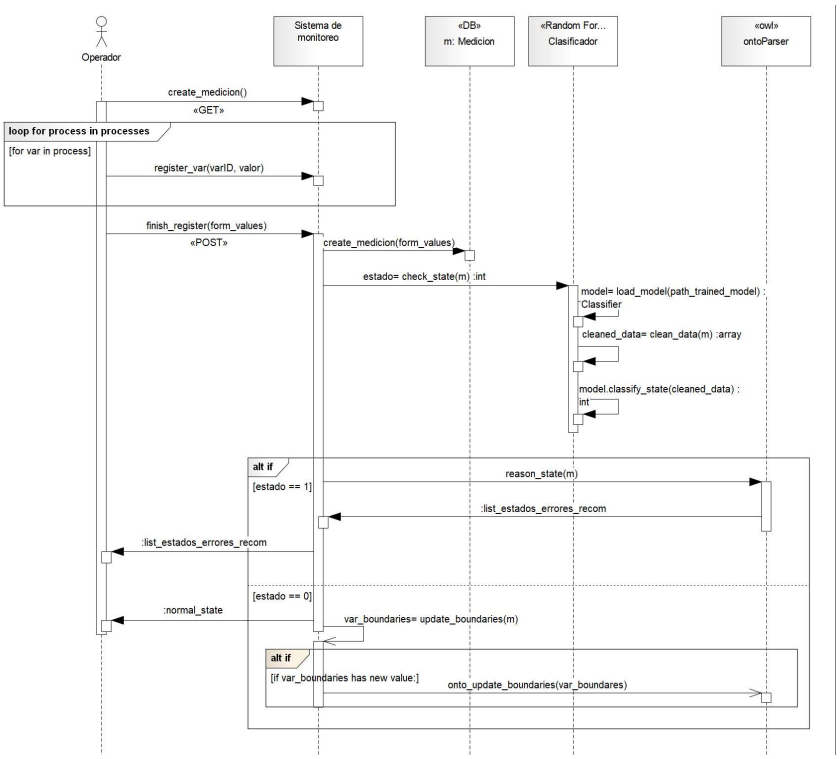

Figura E.1: Diagrama de secuencia: agregar medición

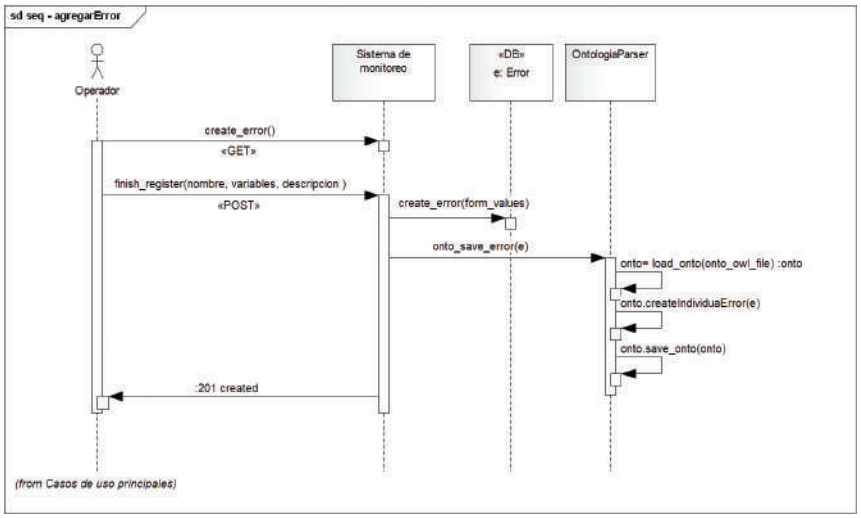

Figura E.2: Diagrama de secuencia: agregar error

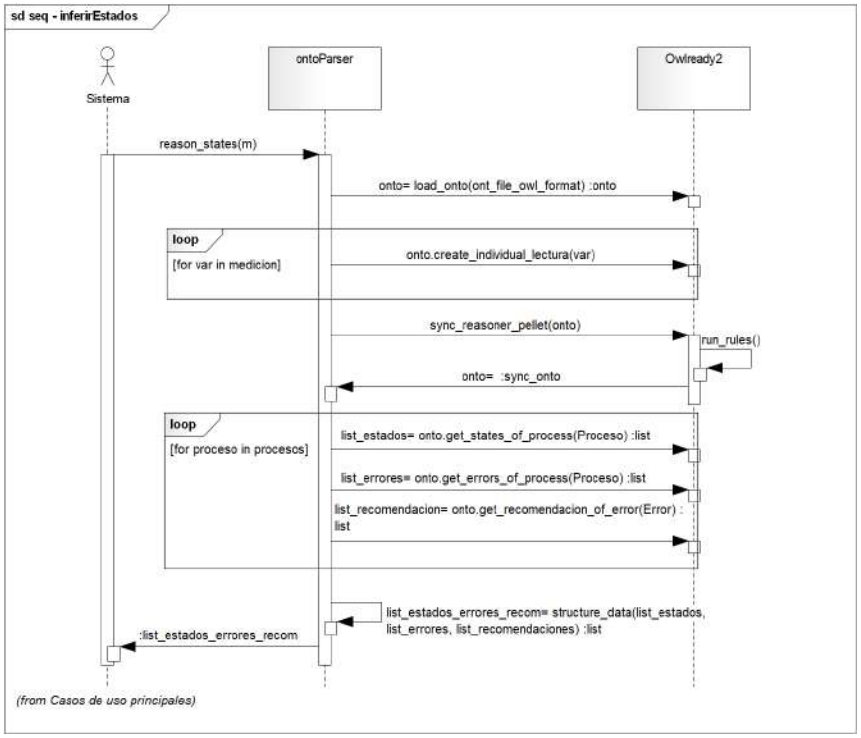

Figura E.3: Diagrama de secuencia: inferir estados

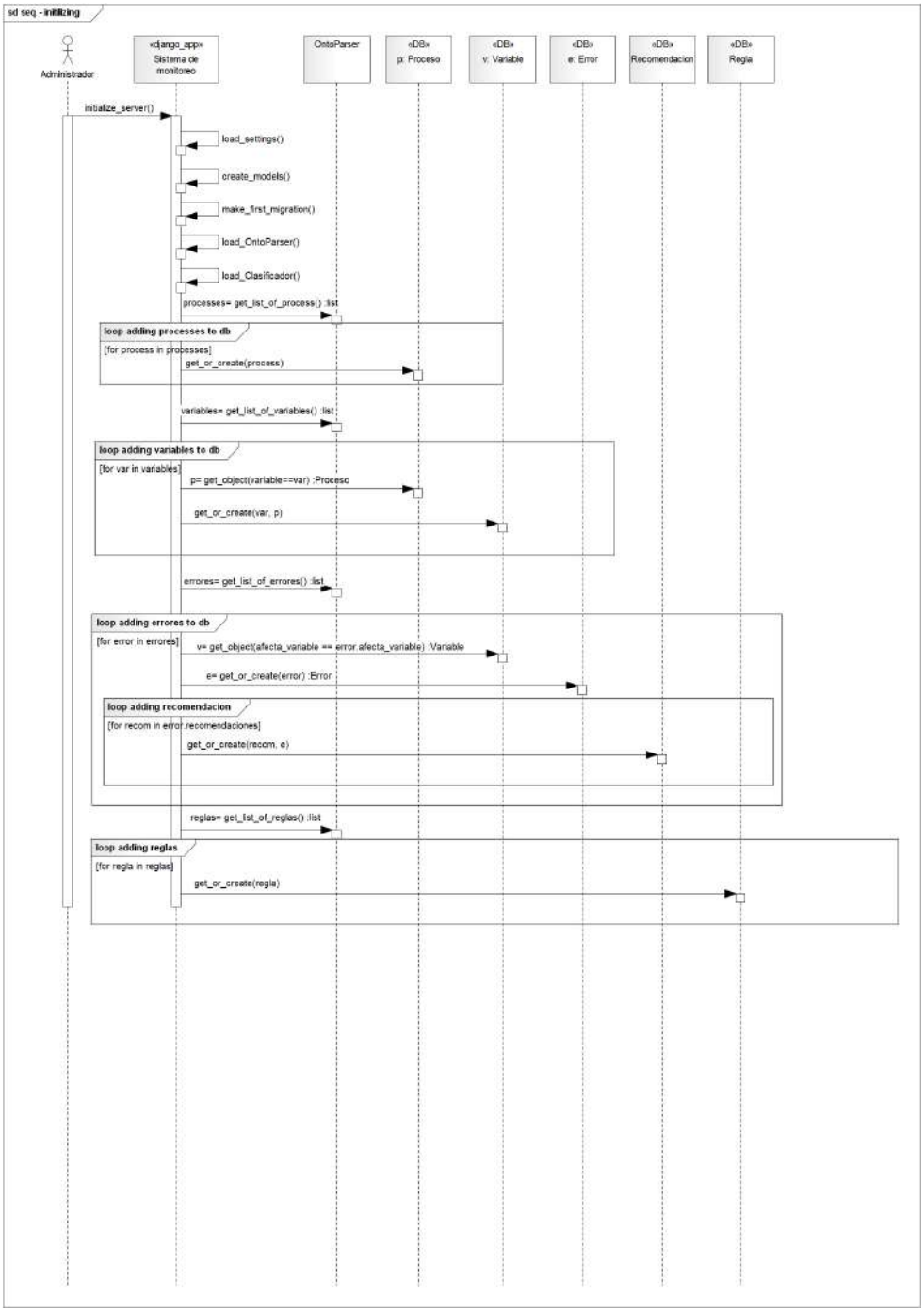

Figura E.4: Diagrama de secuencia: inicialización

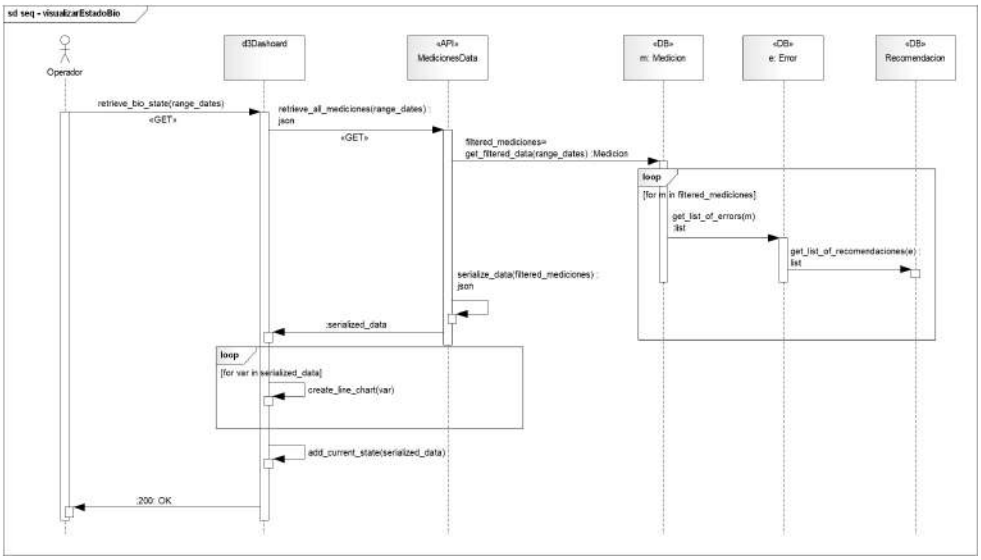

Figura E.5: Diagrama de secuencia: visualizar estados

Apéndice F

Django - Modelos

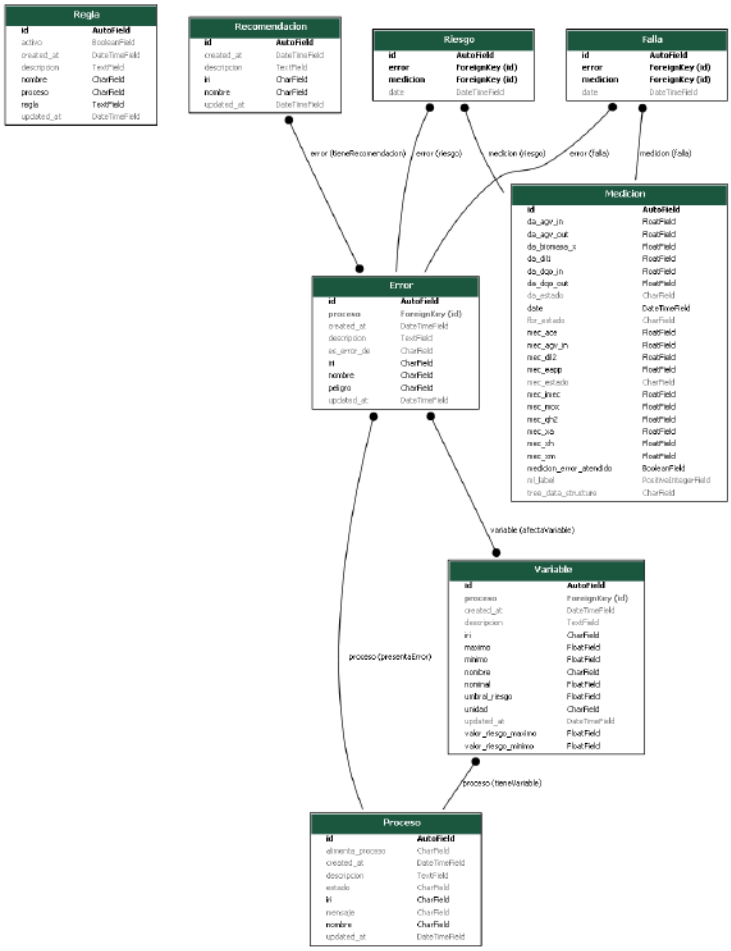

Figura F.1: Modelos para base de datos usados en Django

Bibliografía

[1] M. J. Friedel, "Environmental Modelling & Software Data-driven modeling of surface temperature anomaly and solar activity trends," *Environmental Modelling and Software*, vol. 37, pp. 217–232, 2012.

[2] J. Rabatel, S. Bringay, and P. Poncelet, "Anomaly detection in monitoring sensor data for preventive maintenance," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7003–7015, 2011.

[3] R. Ul Islam, M. S. Hossain, and K. Andersson, "A novel anomaly detection algorithm for sensor data under uncertainty," *Soft Computing*, vol. 22, no. 5, pp. 1623–1639, 2018.

[4] K. Peerbhay, O. Mutanga, R. Lottering, and R. Ismail, "Unsupervised anomaly weed detection in riparian forest areas using hyperspectral data and LiDAR," *Workshop on Hyperspectral Image and Signal Processing, Evolution in Remote Sensing*, 2017.

[5] H. Al-Thani, H. Hassen, S. Al-Maadced, N. Fetais, and A. Jaoua, "Unsupervised Technique for Anomaly Detection in Qatar Stock Market," *2018 International Conference on Computer and Applications (ICCA)*, pp. 116–9, 2018.

[6] Z. Zhao, M. G. Kishan, and M. K. Chilukuri, "Online Anomaly Detection Using Random Forest," in *Recent Trends and Future Technology in Applied Intelligence*, pp. 135–147, 2018.

[7] M. Kopp, T. Pevný, and M. Holeňa, "Anomaly explanation with random forests," *Expert Systems with Applications*, vol. 149, 2020.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *THE DESCRIPTION LOGIC HANDBOOK*. 2003.

[9] BP p.l.c. 2019, "BP Energy Outlook 2019 edition," *BP Energy Outlook 2019*, 2019.

[10] B. Kamm, P. R. Gruber, and M. Kamm, *Biorefineries – Industrial Processes and Products*, vol. 1. 2006.

[11] R. Julio, J. Albet, C. Vialle, C. Vaca-Garcia, and C. Sablayrolles, "Sustainable design of biorefinery processes: existing practices and new methodology," *Biofuels, Bioproducts and Biorefining*, vol. 11, no. 2, pp. 373–395, 2017.

[12] Z. Bubnicki, *Modern Control Theory*. Springer Berlin Heidelberg, 2005.

[13] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Prentice Hall, twelfth ed ed., 2011.

[14] Transport Safety Board of Canada, *Railway Investigation Report R13D0054. Runaway*

and main-track derailment. Montreal, Maine & Atlantic Railway Freight train MMA-002 Mile 0.23, Sherbrooke Subdivision Lac-Mégantic, Quebec 06 July 2013. No. July, 2013.

[15] F. Balkau, "Learning from Baia Mare," 2005.

[16] W. Giger, "The Rhine red, the fish dead-the 1986 Schweizerhalle disaster, a retrospect and long-term impact assessment," *Environmental Science and Pollution Research*, vol. 16, no. SUPPL1, pp. 98–111, 2009.

[17] THE REGIONAL and ENVIRONMENTAL CENTER, "The Cyanide Spill at Baia Mare , Romania," *Report*, p. 8, 2000.

[18] S. X. Ding, *Model-based fault diagnosis techniques: Design schemes, algorithms, and tools*. 2008.

[19] L. Wang and X. Deng, "Multi-block principal component analysis based on variable weight information and its application to multivariate process monitoring," *Canadian Journal of Chemical Engineering*, vol. 96, no. 5, pp. 1127–1141, 2018.

[20] Y. Dong and S. J. Qin, "A novel dynamic PCA algorithm for dynamic data modeling and process monitoring," *Journal of Process Control*, vol. 67, pp. 1–11, 2018.

[21] Z. Chen, S. X. Ding, H. Luo, and K. Zhang, "An alternative data-driven fault detection scheme for dynamic processes with deterministic disturbances," *Journal of the Franklin Institute*, vol. 354, no. 1, pp. 556–570, 2017.

[22] G. Zhiqiang, "Review on data-driven modeling and monitoring for plant-wide industrial processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 171, no. 4, pp. 16–25, 2017.

[23] S. Joe Qin, *Data-driven fault detection and diagnosis for complex industrial processes*, vol. 42. IFAC, 2009.

[24] S. Gajjar, M. Kulahci, and A. Palazoglu, "Real-time fault detection and diagnosis using sparse principal component analysis," *Journal of Process Control*, vol. 67, pp. 112–128, 2018.

[25] S. Gajjar and A. Palazoglu, "A data-driven multidimensional visualization technique for process fault detection and diagnosis," *Chemometrics and Intelligent Laboratory Systems*, vol. 154, pp. 122–136, 2016.

[26] M. E. L. Koujok and M. Amazouz, "Data-Driven algorithms for fault detection and diagnosis in industrial process," *Int'l Conf. on Advances in Big Data Analytics*, pp. 115–116, 2016.

[27] H. Wu and J. Zhao, "Deep convolutional neural network model based chemical process fault diagnosis," *Computers and Chemical Engineering*, vol. 115, pp. 185–197, 2018.

[28] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault Diagnosis in Chemical Processes Using Fisher Discriminant Analysis , Discriminant Partial Least Squares , and Principal Component Analysis," pp. 243–252, 2000.

[29] G. Li, S. J. Qin, and T. Yuan, "Data-driven root cause diagnosis of faults in process industries," *Chemometrics and Intelligent Laboratory Systems*, vol. 159, no. September, pp. 1–11, 2016.

[30] C.-d. Tong, X.-f. Yan, and Y.-x. Ma, "Statistical process monitoring based on improved principal component analysis and its application to chemical processes," *Journal of*

Zhejiang University SCIENCE A, vol. 14, no. 7, pp. 520–534, 2013.

[31] C. L. S. Rajasegarar and M. Palaniswami, "[Survey] Anomaly Detection in Wireless Sensor Networks," *Ieee Wireless Communications*, no. August, pp. 34–40, 2008.

[32] V. Alcaraz González, "Modelado y optimización de una biorefinería para la producción de hidrógeno a partir de aguas residuales usando celdas electrolíticas microbianas y el aprovechamiento de subproductos para el cultivo y revalorización de microalgas," vol. 19, no. 1, 2016.

[33] J. MacGregor and A. Cinar, "Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods," *Computers and Chemical Engineering*, vol. 47, pp. 111–120, 2012.

[34] C. J. Gommes, S. Blacher, J. H. Dunsmuir, and A. H. Tsou, "Practical Methods for Measuring the Tortuosity of Porous Materials from Binary or Gray-Tone Tomographic Reconstructions," *AIChE Journal*, vol. 55, no. 8, pp. 2000–2012, 2012.

[35] Z. Ge, Z. Song, and F. Gao, "Review of recent research on data-based process monitoring," *Industrial & Engineering Chemistry Research*, vol. 52, no. 10, pp. 3543–3562, 2013.

[36] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[37] G. Tchobanoglous, F. Burton, H. Stensel, I. Metcalf & Eddy, and F. Burton, *Wastewater Engineering: Treatment and Reuse*. McGraw-Hill higher education, McGraw-Hill Education, 2003.

[38] B. E. Logan and J. M. Regan, "Microbial fuel cells - Challenges and applications," *Environmental Science and Technology*, vol. 40, no. 17, pp. 5172–5180, 2006.

[39] C. I. Torres, "On the importance of identifying, characterizing, and predicting fundamental phenomena towards microbial electrochemistry applications," *Current Opinion in Biotechnology*, vol. 27, pp. 107–114, 2014.

[40] G. Cea-Barcia, G. Buitrón, G. Moreno, and G. Kumar, "A cost-effective strategy for the bio-prospecting of mixed microalgae with high carbohydrate content: Diversity fluctuations in different growth media," *Bioresource Technology*, vol. 163, pp. 370–373, 2014.

[41] L. Meier, R. Pérez, L. Azócar, M. Rivas, and D. Jeison, "Photosynthetic CO2 uptake by microalgae: An attractive tool for biogas upgrading," *Biomass and Bioenergy*, vol. 73, pp. 102–109, 2015.

[42] J. d. J. Colín Robles, "Simulacion del digestor anaerobio acoplado a una celda de electrolisis microbiana," tech. rep.

[43] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django*. 2009.

[44] J. Ladyman, J. Lambert, and K. Wiesner, *What is a complex system?*, vol. 3. 2013.

[45] Wang, *Complex System Maintenance Handbook - Condition-Based Maintenance modelling*. 2008.

[46] N. Boccara, *Modeling Complex Systems*, vol. 43. Springer-Verlag London Ltd., 2007.

[47] M. Svítek, "Towards complex system theory," *Neural Network World*, vol. 25, no. 1, pp. 5–33, 2015.

[48] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabasi, "Observability of complex systems.(APPLIED PHYSICAL SCIENCES: BIOPHYSICS AND COMPUTATIONAL BIOLOGY)(Author abstract)," *Proceedings of the National Academy of Sciences of the United States*, vol. 110, no. 7, p. 2460, 2013.

[49] H. Hyötyniemi, *COMPLEX SYSTEMS : SCIENCE AT THE EDGE OF CHAOS*. Heikki Hyötyniemi, 2003.

[50] H. Simon, "The Architecture of Complexity," *Proceedings ofthe American Philosophical Society*, vol. 106, no. 6, pp. 467–482, 1962.

[51] J. Kwapień and S. Drozdz, "Physical approach to complex systems," *Physics Reports*, vol. 515, no. 3-4, pp. 115–226, 2012.

[52] S. Ternov and R. Akselsson, "System weaknesses as contributing causes of accidents in health care," *International Journal for Quality in Health Care*, vol. 17, no. 1, pp. 5–13, 2005.

[53] N. G. Leveson, *Systemic Factor in Software-Related Spacecraft Accidents*. PhD thesis, Massachusetts Institute of Technology, 2554.

[54] L. H. Chiang, M. E. Kotanchek, and A. K. Kordon, "Fault diagnosis based on Fisher discriminant analysis and support vector machines," *Computers and Chemical Engineering*, vol. 28, no. 8, pp. 1389–1401, 2004.

[55] H. Wang, A. Nobakhti, and A. Hussain, "Advances in complex control systems theory and applications," *IET Control Theory & Applications*, vol. 4, no. 2, pp. 173–175, 2010.

[56] A. Ayadi, O. Ghorbel, A. M. Obeid, and M. Abid, "Outlier detection approaches for wireless sensor networks: A survey," *Computer Networks*, vol. 129, pp. 319–333, 2017.

[57] A. I. Zecevic and D. D. Siljak, *Control of Complex Systems: Structural Constraints and Uncertainty*. Springer New York Dordrecht Heidelberg London, 2010.

[58] C. L.H., R. E.L, and B. R.D, *Fault Detection and Diagnosis in Industrial Systems*. Springer-Verlag London Ltd., 2002.

[59] V. G. Kaburlasos, *Towards a Unified Modeling and Knowledge- Representation based on Lattice Theory*. 2006.

[60] A. M. Ertiame, D. Yu, F. Yu, and J. Gomm, "Robust fault diagnosis for an exothermic semi-batch polymerization reactor under open-loop," *Systems Science & Control Engineering*, vol. 3, no. 1, pp. 14–23, 2015.

[61] H. Dong, Z. Wang, S. X. Ding, and H. Gao, "Event-based h_∞ filter design for a class of nonlinear time-varying systems with fading channels and multiplicative noises," *IEEE Transactions on Signal Processing*, vol. 63, pp. 3387–3395, July 2015.

[62] H. Dong, Z. Wang, S. X. Ding, and H. Gao, "On h_∞ estimation of randomly occurring faults for a class of nonlinear time-varying systems with fading channels," *IEEE Transactions on Automatic Control*, vol. 61, pp. 479–484, Feb 2016.

[63] B. Shen, Z. Wang, and Y. Hung, "Distributed h-consensus filtering in sensor networks with multiple missing measurements: The finite-horizon case," *Automatica*, vol. 46, no. 10, pp. 1682 – 1688, 2010.

[64] A. Boussif and M. Ghazel, "Model-Based Monitoring of a Train Passenger Access System," *IEEE Access*, vol. 6, pp. 41619–41632, 2018.

[65] P. Frank and X. Ding, "Survey of robust residual generation and evaluation methods in observer-based fault detection systems," *Journal of Process Control*, vol. 7, no. 6, pp. 403 – 424, 1997.

[66] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance W*, vol. 91. Springer US, 2017.

[67] T. Kim, Y. Wang, H. Fang, Z. Sahinoglu, T. Wada, S. Hara, and W. Qiao, "Model-based condition monitoring for lithium-ion batteries," *Journal of Power Sources*, vol. 295, pp. 16–27, 2015.

[68] A. Bakdi and A. Kouadri, "A new adaptive PCA based thresholding scheme for fault detection in complex systems," *Chemometrics and Intelligent Laboratory Systems*, vol. 162, no. December 2016, pp. 83–93, 2017.

[69] D. Xiao, J. Jiang, Y. Mao, and X. Liu, "Process Monitoring and Fault Diagnosis for Piercing Production of Seamless Tube," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–13, 2016.

[70] S. Misra and M. Nikolaou, "A data-driven modeling approach to zonal isolation of cemented gas wells," *Journal of Natural Gas Science and Engineering*, vol. 59, no. August, pp. 262–273, 2018.

[71] S. X. Ding, *Data-driven Design of Fault Diagnosis and Fault-tolerant Control Systems*. 2014.

[72] Z. Ge, C. Yang, and Z. Song, "Improved kernel PCA-based monitoring approach for nonlinear processes," *Chemical Engineering Science*, vol. 64, no. 9, pp. 2245–2255, 2009.

[73] R. Dunia, G. Rochelle, T. F. Edgar, and M. Nixon, "Multivariate monitoring of a carbon dioxide removal process," *Computers and Chemical Engineering*, vol. 60, pp. 381–395, 2014.

[74] J. Li and P. Cui, "Improved kernel fisher discriminant analysis for fault diagnosis," *Expert Systems with Applications*, vol. 36, no. 2 PART 1, pp. 1423–1432, 2009.

[75] J. Yu, "Nonlinear bioprocess monitoring using multiway kernel localized fisher discriminant analysis," *Industrial and Engineering Chemistry Research*, vol. 50, no. 6, pp. 3390–3402, 2011.

[76] J. Tessier, C. Duchesne, G. P. Tarcy, C. Gauthier, and G. Dufour, "Multivariate analysis and monitoring of the performance of aluminum reduction cells," *Industrial and Engineering Chemistry Research*, vol. 51, no. 3, pp. 1311–1323, 2012.

[77] X. B. He, W. Wang, Y. P. Yang, and Y. H. Yang, "Variable-weighted Fisher discriminant analysis for process fault diagnosis," *Journal of Process Control*, vol. 19, no. 6, pp. 923–931, 2009.

[78] D. Ramotsoela, A. Abu-Mahfouz, and G. Hancke, "A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study," *Sensors (Switzerland)*, vol. 18, no. 8, pp. 1–25, 2018.

[79] V. J. Hodge and J. I. M. Austin, "A Survey of Outlier Detection Methodologies," no. 1969, pp. 85–126, 2004.

[80] J. Fan and Y. Wang, "Fault detection and diagnosis of non-linear non-Gaussian dynamic processes using kernel dynamic independent component analysis," *Information*

Sciences, vol. 259, pp. 369–379, 2014.

[81] J. Gong and F. You, "You paper," vol. 60, no. 9, 2014.

[82] J.-M. Lee, S. J. Qin, and I.-B. Lee, "Fault Detection of Non-Linear Processes Using Kernel Independent Component Analysis," *The Canadian Journal of Chemical Engineering*, vol. 85, no. 4, pp. 526–536, 2010.

[83] J. Y. Fan, M. Nikolaou, and R. E. White, "An approach to fault diagnosis of chemical processes via neural networks," *AIChE Journal*, vol. 39, no. 1, pp. 82–88, 1993.

[84] Y. Tinghu, Z. Binglin, and H. Ren, "A neural network methodology for rotating machinery fault diagnosis," *Tooldiag'93, Int. Conf. on Fault Diagnosis*, vol. 35, no. 12, pp. 170–178, 1993.

[85] I. Yélamos, G. Escudero, M. Graells, and L. Puigjaner, "Performance assessment of a novel fault diagnosis system based on support vector machines," *Computers and Chemical Engineering*, vol. 33, no. 1, pp. 244–255, 2009.

[86] A. Kulkarni, V. K. Jayaraman, and B. D. Kulkarni, "Knowledge incorporated support vector machines to detect faults in Tennessee Eastman Process," *Computers and Chemical Engineering*, vol. 29, no. 10, pp. 2128–2133, 2005.

[87] S. Mahadevan and S. L. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *Journal of Process Control*, vol. 19, no. 10, pp. 1627–1639, 2009.

[88] Y. Zhang, "Independent Component Analysis (KICA) and Support Vector Machine (SVM)," *Industrial & Engineering Chemistry Research*, vol. 47, pp. 6961–6971, 2008.

[89] T. C. Wu and M. F. Hsu, "Credit risk assessment and decision making by a fusion approach," *Knowledge-Based Systems*, vol. 35, pp. 102–110, 2012.

[90] s. Karakurt, S. Özer, T. Ulusinan, and M. C. Ganiz, "A machine learning approach to database failure prediction," *2nd International Conference on Computer Science and Engineering, UBMK 2017*, pp. 1030–1035, 2017.

[91] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest," *Security and Communication Networks*, vol. 2018, 2018.

[92] G. Louppe, "Understanding Random Forests: From Theory to Practice," no. July, 2014.

[93] A. Criminisi and J. Shotton, *Decision Forests for Computer Vision and Medical Image Analysis*. Springer-Verlag London, 2013.

[94] K. Ayyadevara, *Pro machine learning algorithms : a hands-on approach to implementing algorithms in Python and R*. 2018.

[95] G. Williams, *Data Mining with Rattle and R*. 2011.

[96] L. Breiman, "Random forests," *Random Forests*, pp. 1–122, 2001.

[97] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random Forests," in *Ensemble Machine Learning*, pp. 157–175, 2012.

[98] S. D. Anton, S. Sinha, and H. Dieter Schotten, "Anomaly-based intrusion detection in industrial data with SVM and random forests," *2019 27th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2019*, 2019.

[99] E. Russell, L. H. Chiang, and R. D. Braatz, *Data-driven Methods for Fault Detection and Diagnosis in Chemical Processes*. 2000.
[100] M. K. Bergman, *A Knowledge Representation Practionary*. Springer Nature Switzerland AG, 2018.
[101] M. Chein and M.-L. Mugnier, *Graph-based Knowledge Representation Computational*, vol. 53. 2013.
[102] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*. 2004.
[103] S. Grimm, P. Hitzler, and A. Abecker, "Knowledge Representation and Ontologies Logic, Ontologies and Semantic Web Languages," no. September, 2007.
[104] J. Reyes-Ortiz, A. Jiménez, and J. Cater, "Ontology-based Knowledge Representation for Supporting Medical Decisions," *Advances in Soft . . .*, vol. 68, no. 2013, pp. 127–136, 2013.
[105] G. Jakus, V. Milutinović, S. Omerović, and S. Tomažič, *Concepts, ontologies, and knowledge representation*. 2013.
[106] Y. Lin, *Semantic annotation for process models: Facilitating Process Knowledge Management via Semantic Interoperability*. No. 2008:3, 2008.
[107] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and methods," *Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.
[108] R. Hoekstra, "Ontology representation: Design patterns and ontologies that make sense," *Frontiers in Artificial Intelligence and Applications*, vol. 197, no. 1, pp. 1–236, 2009.
[109] G. Antoniou and F. Van Harmelen, "Web ontology languages," *Semantic Web Services: Theory, Tools and Applications*, pp. 96–109, 2007.
[110] M. Merdan, M. Vallee, W. Lepuschitz, and A. Zoitl, "Monitoring and diagnostics of industrial systems using automation agents," *International Journal of Production Research*, vol. 49, no. 5, pp. 1497–1509, 2011.
[111] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis part III: Process history based methods," *Computers and Chemical Engineering*, vol. 27, no. 3, pp. 327–346, 2003.
[112] M. B. Jain, M. B. Srinivas, and A. Jain, "A novel web based expert system architecture for on-line and off-line fault diagnosis and control (FDC) of power system equipment," *2008 Joint International Conference on Power System Technology POWERCON and IEEE Power India Conference, POWERCON 2008*, 2008.
[113] F. Xu, X. Liu, W. Chen, C. Zhou, and B. Cao, "Ontology-based method for fault diagnosis of loaders," *Sensors (Switzerland)*, vol. 18, no. 3, 2018.
[114] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, S. Jupp, G. Moulton, N. Drummond, and S. Brandt, "A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools," *Matrix*, pp. 0–107, 2011.
[115] "Swrl: A semantic web rule language combining owl and ruleml," 2020.
[116] Antoniou, Grigoris and F. Van Harmelen, *A Semantic Web Primer*. 2008.
[117] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.