UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

DESARROLLO E IMPLEMENTACIÓN DE MODELOS DE APRENDIZAJE DE MÁQUINAS PARA IDENTIFICACIÓN DE FALLAS EN EL MONITOREO DE LA CONDICIÓN DE ACTIVOS EN PLANTA CELULOSA NUEVA ALDEA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

LUCAS IGNACIO ALARCÓN GONZÁLEZ

PROFESORA GUÍA:
VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:
MARCOS GONZÁLEZ VALLEJOS
BENJAMIN HERRMANN PRIESNITZ

Este trabajo ha sido parcialmente financiado por Celulosa Arauco y Constitución S.A.

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: LUCAS IGNACIO ALARCÓN GONZÁLEZ
FECHA: 2022
PROF. GUÍA: VIVIANA MERUANE NARANJO

DESARROLLO E IMPLEMENTACIÓN DE MODELOS DE APRENDIZAJE DE MÁQUINAS PARA IDENTIFICACIÓN DE FALLAS EN EL MONITOREO DE LA CONDICIÓN DE ACTIVOS EN PLANTA CELULOSA NUEVA ALDEA

En este trabajo de título se desarrollan modelos de identificación de fallas basados en algoritmos de Aprendizaje de Máquinas, para el monitoreo de activos en Planta de Celulosa Nueva Aldea, con la finalidad de detectar de forma oportuna averías en el parque de equipos.

Para lograr esto, se estudiaron diferentes propuestas de equipos y finalmente fueron seleccionados 2 casos de estudio. El primero corresponde al desarrollo e implementación de modelos no-supervisados de detección de novedades en familias de turbomáquinas hidráulicas generadoras, con la finalidad de definir una región de operación normal en el espacio de las variables de operación y alertar cuando los equipos operan de forma anómala. El segundo caso corresponde al desarrollo de modelos supervisados de detección temprana de eventos de alto torque en las prensas de desplazamiento de la Línea de Fibra.

Los modelos son desarrollados mediante programación en lenguaje *Python* utilizando la plataforma de acceso libre y gratuita *Google Colaboratory*. Los datos de monitoreo corresponden a variables de operación de cada equipo, que son extraídos desde la plataforma *Wedge* que luego son procesados, estructurados y agrupados en conjuntos de entrenamiento, validación y testeo.

Los resultados obtenidos en el caso turbomáquinas señalan que, entre los modelos evaluados para detección de novedades, los algoritmos *Elliptic Envelope*, *Local Outlier Factor* y *HDBSCAN* tienen un mejor rendimiento en esta aplicación particular, donde el último de los mencionados destaca sobre el resto siendo capaz de definir una frontera de decisión que se ajusta bien a la forma de los datos sin presentar discontinuidades.

Los resultados obtenidos en el caso prensas señalan que los modelos basados en redes neuronales *MLP* son capaces de identificar los peaks de torque y otros comportamientos anómalos en la operación, pero no son capaces de detectar con anticipación estos eventos.

Los modelos basados en redes neuronales recurrentes *LSTM* y redes neuronales convolucionales *CNN*, permiten reconocer patrones temporales y son capaces de predecir eventos de alto torque con hasta 60 minutos de anticipación. Sin embargo, los modelos desarrollados están altamente sobreajustados a los datos de entrenamiento y no tienen capacidad de generalización, mostrando un bajo desempeño predictivo en datos no conocidos.

Tabla de Contenido

1. Introducción

La historia industrial ha estado marcada por distintos hitos que cambiaron radicalmente la forma de producir, aumentando los niveles de producción notablemente, desde la primera revolución industrial con la máquina a vapor, pasando por la electrificación de los procesos productivos hasta llegar a la automatización de estos. Actualmente la integración de los sistemas físicos con los digitales, la conectividad *online*, la masificación de los sistemas de información y la creciente utilización de sensores y dispositivos en la industria, indican la presencia de una nueva revolución industrial.

La Cuarta Revolución Industrial o Industria 4.0 cuenta con una serie de tecnologías y herramientas que mejoran la eficiencia y el control de los procesos productivos. Particularmente herramientas como el Internet Industrial de las Cosas (*IIoT*), el Análisis de Datos y la Inteligencia Artificial (*AI*), son utilizadas para tareas tan importantes en la industria como el Mantenimiento. La presencia de sensores no solo permite el monitoreo continuo y en tiempo real de los activos, sino también la recopilación de enormes cantidades de datos que pueden ser usados para entrenar modelos basados en algoritmos de Aprendizaje de Máquinas (una rama de la Inteligencia Artificial), con el objetivo de desarrollar modelos de detección, diagnóstico y pronóstico de fallas.

El presente trabajo se enmarca en la Planta de Celulosa Nueva Aldea de la empresa Celulosa Arauco y Constitución, la cual cuenta con un parque de más de 36.000 equipos, cada uno de los cuales juega un rol diferente en el proceso de producción de Celulosa. En términos del negocio, es relevante para la compañía el maximizar el beneficio obtenido a partir de sus activos, donde parte de los costos de producción están asociados al mantenimiento y a las pérdidas productivas debido a la indisponibilidad de los activos. Para optimizar el proceso de mantenimiento es clave poder identificar con anticipación aquellos equipos con síntomas de falla, permitiendo planificar las respectivas actividades de mantenimiento con el menor impacto posible en la disponibilidad de los activos críticos. Por tanto, este trabajo de título tiene por finalidad desarrollar e implementar modelos de identificación de fallas para el monitoreo de la condición de activos en la Planta de Celulosa Nueva Aldea.

1.1. Objetivos y Alcances

1.2. Objetivo General

Desarrollar e implementar modelos de identificación de fallas basados en algoritmos de Aprendizaje de Máquinas, para el monitoreo de activos en Planta de Celulosa Nueva Aldea, con la finalidad de detectar de forma oportuna averías en el parque de equipos.

1.3. Objetivos Específicos

- Seleccionar uno o varios equipos de la planta para su estudio.
- Desarrollar e implementar modelos no supervisados de detección de anomalías en familia de Turbomáquinas.
- Desarrollar modelos de detección temprana de alto torque en Prensas de desplazamiento.
- Validar y evaluar los modelos desarrollados.
- Reportar los resultados al área de confiabilidad de la Planta Nueva Aldea.

1.4. Alcances

Los alcances del presente trabajo son los siguientes:

- Los modelos desarrollados están basados en algoritmos de *Machine Learning* y *Deep Learning.*
- Los modelos desarrollados corresponden a modelos de detección de anomalías y modelos de detección temprana de eventos de alto torque.
- Los algoritmos utilizados han sido reportados en la literatura para aplicaciones similares a las aquí estudiadas. Si bien se ajustaron elementos de la arquitectura de algunos modelos, este trabajo no se enfoca en el desarrollo de nuevos algoritmos, sino en la aplicación de estos algoritmos a los objetivos específicos antes planteados.

- Los datos utilizados para el desarrollo de los modelos se acotan a las variables de monitoreo de los equipos de la planta Nueva Aldea almacenados en la plataforma *Wedge*, e información proporcionada por Arauco.
- Los equipos fueron seleccionados en base a criterios como la disponibilidad de los datos de monitoreo y/o registros históricos de fallas/eventos, la relevancia del equipo en el proceso productivo, entre otros aspectos.
- Otro criterio de selección es la calidad de los datos, para ello se realizó un análisis preliminar de los datos donde se verificó que las mediciones estuvieran dentro de los rangos de operación y que efectivamente existan mediciones en el periodo de interés.
- Si bien se realizaron varias iteraciones, los modelos desarrollados no están optimizados y esto queda propuesto como trabajo futuro.
- Los modelos desarrollados y los resultados que aquí se presentan fueron programados en lenguaje *python.* Se utilizó la plataforma *Google Colaboratory* de acceso libre y gratuito, cuyos servidores tienen una capacidad de 13 GB de memoria RAM y 108 GB de almacenamiento.

2. Antecedentes

2.1. Planta de Celulosa Nueva Aldea

El Complejo Forestal Industrial Nueva Aldea ubicado en la Región del Ñuble, al sur de la intersección de los ríos Ñuble e Itata, pertenece a la empresa Celulosa Arauco y Constitución (ARAUCO) dedicada a la fabricación de Pulpa de Celulosa y derivados como maderas aserradas y paneles. La Planta de Celulosa Nueva Aldea (PCNA) consolidó a ARAUCO como una de las compañías forestales e industriales más importantes a nivel mundial. La construcción de la Planta se llevó a cabo en 2 Fases: La Fase I, llevada a cabo en el año 2004 requirió una inversión de 252 Millones de dólares para la producción de Paneles y Aserradero, incluyendo la Planta de Generación de Energía con una capacidad de 210 ton/h y 29MW. La Fase II, en el año 2006, contempló la construcción de 2 Líneas de producción de Celulosa (para Pino y Eucaliptus) con una inversión de 850 Millones de dólares. El 31 de agosto de 2006 se produce el primer fardo de celulosa de Eucaliptus. En 2010 se realizaron mejoras y modificaciones al proceso incrementando la capacidad de producción de 856.000 ADt/año (Toneladas de Celulosa Seca en el Aire al año) a 1 Millón de ADt/año, en tanto la capacidad de Generación de Energía ahora es de 140MW, con una capacidad de Entrega al SEN (Sistema Eléctrico Nacional) de 50 MW [1].

2.2. Proceso productivo

El proceso productivo de la celulosa es bastante complejo y difícil de describir en detalle, por tanto, una descripción general bastará para poner en contexto el funcionamiento de la PCNA. El proceso principal es el que transforma las astillas de madera en fardos de celulosa, y es conocido como *Línea de Fibra*. El proceso comienza en el *Área de Madera* con la recepción, almacenamiento y preparación de la madera donde los troncos de Pino y Eucaliptus son descortezados y astillados para ser almacenados en pilas. La madera en forma de chip pasa al *Área de Digestores*, donde se incorpora Licor Blanco para la impregnación y cocción de la madera y comenzando la formación de pulpa. Esta pulpa se hace pasar por procesos de Lavado y Deslignificación donde se extrae el Licor Negro de la pulpa. Luego vienen los procesos de Pre-Blanqueo y Blanqueo que incorpora Dióxido de Cloro para otorgarle el color blanco a la pulpa. Posteriormente la pulpa pasa al *Área de Máquina* donde se prepara la formación de una hoja continua de celulosa, pasando por el Secado y Cortado

de la hoja para finalmente pasar al Embalado de fardos de celulosa que son almacenados en la *Bodega de Producto Terminado* hasta que *Logística* gestione su despacho.

En conjunto con el proceso principal existe un *Ciclo de Recuperación de Reactivos y Generación de Energía* encargado de suministrar Insumos y Energía al proceso y a la Planta. El Dióxido de Cloro para los procesos de Pre-Blanqueo y Blanqueo es producido en el *Área Química* usando como insumo el Clorato de Sodio. Esta área también está encargada de almacenar insumos químicos para otras áreas de proceso. En cuanto al Licor Negro, una vez extraído de los Digestores es concentrado en los *Evaporadores* para luego ser combustionado en la *Caldera Recuperadora* y producir Vapor y Licor Verde. Este Licor Verde es llevado al *Área de Caustificación* donde reacciona con Cal para producir Licor Blanco y Lodo. La Cal es producida en el *Horno de Cal* a partir de la calcinación de Carbonato de Calcio (contenido en el Lodo). Además, la Planta cuenta con plantas de *Tratamiento de Aguas y Efluentes* para la producción de agua industrial y tratamiento de RILES. La generación de Energía se realiza en el *Área de Turbo Generadores* donde dos Turbinas de Vapor son alimentadas desde la *Caldera Recuperadora* y un tercera Turbina es alimentada por la *Caldera de Poder*. Esta última utiliza Biomasa como combustible, proveniente del *Área de Madera*.

2.3. Equipos estudiados

El proceso de producción de celulosa se trata de un proceso continuo donde cualquier detención afecta en forma directa los niveles de producción. En cada sub-proceso existen equipos que resultan ser críticos y por tanto es importante asegurar un funcionamiento continuo y predecible de éstos. A continuación, se describen los equipos seleccionados como casos de aplicación para el desarrollo de modelos de Aprendizaje de Máquinas.

2.3.1. Turbomáquinas

Las turbomáquinas son un tipo de máquina hidráulica. Según [2] una máquina hidráulica se define como aquella máquina en que el fluido, o bien proporciona la energía que absorbe la máquina, o bien recibe la energía entregada por la máquina en forma de energía mecánica.

Las máquinas de fluido se pueden clasificar en máquinas hidráulicas o térmicas. En el caso de las hidráulicas, la densidad o volumen específico del fluido no cambia demasiado en su paso por las máquinas y se asume constante, mientras que en las térmicas sí existe una variación significativa.

Por su parte las máquinas hidráulicas pueden clasificarse según el principio de funcionamiento en: máquinas de desplazamiento positivo y turbomáquinas. Las primeras, también conocidas como máquinas volumétricas, el intercambio de energía ocurre en forma de energía de presión creada por la variación de volumen [2]. Por otro lado, en las turbomáquinas el intercambio de energía está asociado a los cambios cantidad de movimiento del fluido.

Por último, las máquinas hidráulicas pueden clasificarse en motoras, cuando el flujo de energía es desde el fluido a la máquina, y generadoras, cuando se restituye energía al fluido.

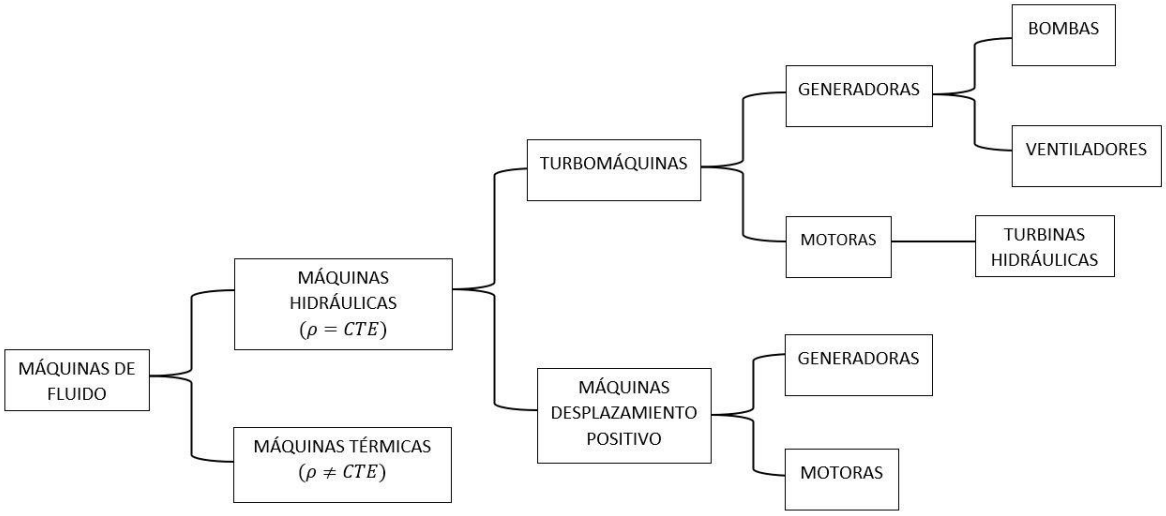


Figura 2.1: Clasificación de máquinas de fluido (Elaboración propia).

Descripción del equipo

En este trabajo, se estudiaron turbomáquinas generadoras, las cuales pueden clasificarse en 2 grupos según el fluido de trabajo que ocupan: Bombas y Ventiladores. Es importante mencionar que este trabajo se enfocó en equipos de velocidad variable.

Bombas

Las bombas se ocupan para impulsar toda clase de líquidos, como agua, aceites de lubricación, químicos reactantes ocupados en el proceso productivo como el licor verde, entre otros. Las bombas que pertenecen al grupo de las turbomáquinas son rotativas y su funcionamiento se basa en la ecuación de Euler, en otras palabras, son rotodinámicas [2].

Pueden ser de flujo radial, axial o una combinación de ambos. Otras características como la posición del eje, presiones de trabajo, número de flujos y rodetes definen el tipo de bomba. Para ilustrar los elementos principales de una bomba, se describe la bomba centrífuga mostrada en la Figura 2.2.

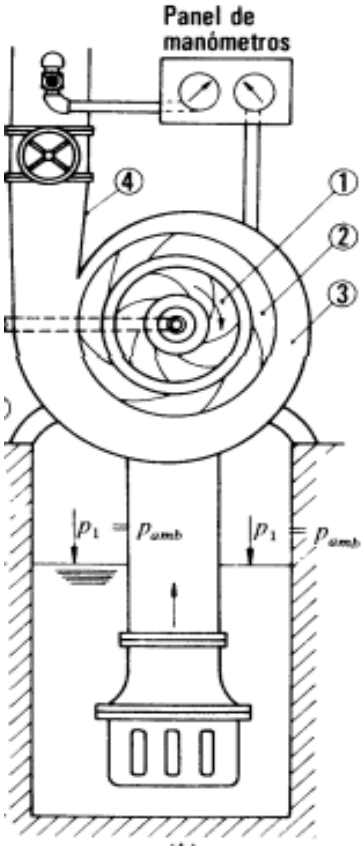


Figura 2.2: Componentes principales de una bomba centrífuga [2].

Los principales componentes de una bomba son: El Rodete (1) que gira solidario con el eje de la máquina, tiene un cierto número de álabes que imparten energía al fluido en forma de energía cinética y presión; La Corona directriz es una corona de álabes fijos que recoge el líquido del rodete y transforma la energía cinética en presión; Una Caja espiral (3) transforma la energía cinética en presión y recoge con pérdidas mínimas el fluido que sale del rodete hasta la tubería de salida; Por último, un Tubo difusor (4) realiza una tercera etapa de difusión, es decir, transforma la energía cinética en presión.

Ventiladores

Un ventilador es en esencia una bomba que impulsa gases, como vapor de agua, gases de combustión, entre otros. Si bien los gases son fluidos compresibles, en el caso de los

ventiladores el incremento de presión es pequeño y por tanto la densidad o el volumen específico no cambian notablemente por tanto es seguro asumir que el gas se comporta como fluido incompresible [2].

Los ventiladores pueden clasificarse según la presión de trabajo en Baja, Media y Alta presión, aunque como se debe destacar que las presiones no superan los 100 mbar. También se pueden diferenciar según el tipo de flujo en Centrífugos y Axiales.

Entre los ventiladores estudiados se tienen inductores, sopladores, extractores y ventiladores de tiro inducido. Para ilustrar los principales componentes de un ventilador, se describe el ventilador de tiro inducido mostrado en la Figura 2.3.

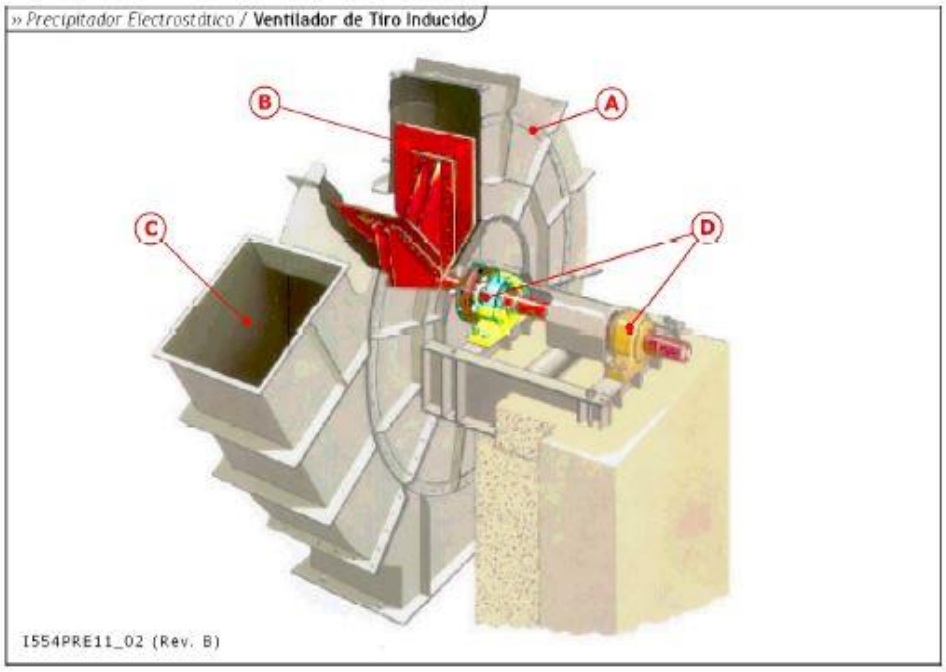


Figura 2.3: Componentes principales de un Ventilador de Tiro Inducido [3].

Los principales componentes del equipo son: La carcasa (A) que protege al impulsor y conduce el gas hacia el ducto de salida; Un Impulsor radial (B) que corresponde al elemento rotor y está conectado al eje del ventilador encargado de transmitir la potencia proveniente de un motor eléctrico generalmente. La potencia del motor es regulada mediante un variador de frecuencia; Un Ducto de salida (C) dirige el gas hacia el siguiente componente, por lo general una chimenea, donde es expulsado al ambiente; Por último, los Descansos (D) soportan el eje del ventilador y absorben las vibraciones durante la operación.

2.3.2. Prensas de desplazamiento

Las prensas de desplazamiento están encargadas del proceso de lavado de la pulpa, donde se retiran elementos como la Lignina y químicos reactantes incorporados en procesos previos. Este proceso de lavado se lleva a cabo en dos áreas de la línea de fibra: el área de pasta café, después de las etapas de clasificación y deslignificación con oxígeno, y el área de blanqueo, después de cada etapa de blanqueo. A continuación, se describe cada etapa de lavado [4].

Etapas de Lavado

i) Lavado pre-oxígeno

La pulpa proveniente de la etapa de clasificación es lavada con licores débiles para retirar elementos disueltos como soda, lignina y jabón, dejando la pulpa en mejores condiciones para la deslignificación con oxígeno.

ii) Lavado post-oxígeno

Se lava la pulpa proveniente de la etapa de deslignificación por oxígeno, retirando elementos como la lignina y así preparar la pulpa para el proceso de blanqueo.

iii) Lavado pre-blanqueo

Se lava la pulpa proveniente de la etapa anterior para retirar contenido de soda residual y ajustar la consistencia de la pulpa para siguientes etapas.

iv) Lavado D0

En esta etapa se lava la pulpa para retirar la lignina previamente disuelta mediante dosificación de Dióxido de Cloro.

v) Lavado Eop

Durante esta etapa se extraen residuales de lignina mediante oxidación con oxígeno y peróxido de hidrógeno, para luego remover los elementos disueltos.

vi) Lavado D1

Se disuelve la lignina aún presente mediante oxidación con Dióxido de Cloro para luego lavar la pulpa y aumentar su blancura.

vii) Lavado D2

Se disuelve la lignina mediante un proceso similar al de la etapa D1 para luego ser retirada en el lavado. La pulpa es enviada a las torres de Alta Densidad Blanca.

Existen diferentes equipos lavadores, entre ellos, filtros, difusores y prensas. Las prensas, o lavadores modernos, utilizan dos principios de lavado: lavado por extracción de agua o desaguado, y lavado por desplazamiento.

Principios de Lavado

i) Desagüe

La extracción de agua o licor ocurre al exprimir la pulpa, de esta forma la consistencia de la pulpa aumenta. Las prensas dependen principalmente de este principio para el lavado de la pulpa.

ii) Desplazamiento

El desplazamiento del agua o licor se produce cuando el líquido de lavado (agua dulce caliente o condensado) empuja el líquido original, de modo que la consistencia de la pulpa se mantiene más o menos constante.

En la planta, las prensas del área de material café se encuentran conectadas en contracorriente, es decir, el flujo de licor de lavado se desplaza en dirección contraria a la de la pulpa, de modo que la cantidad de licor utilizado se minimiza.

Descripción del Equipo

La Figura 2.4 muestra en forma esquemática un corte transversal de una prensa de desplazamiento. A continuación, se describe el funcionamiento del equipo y sus principales componentes.

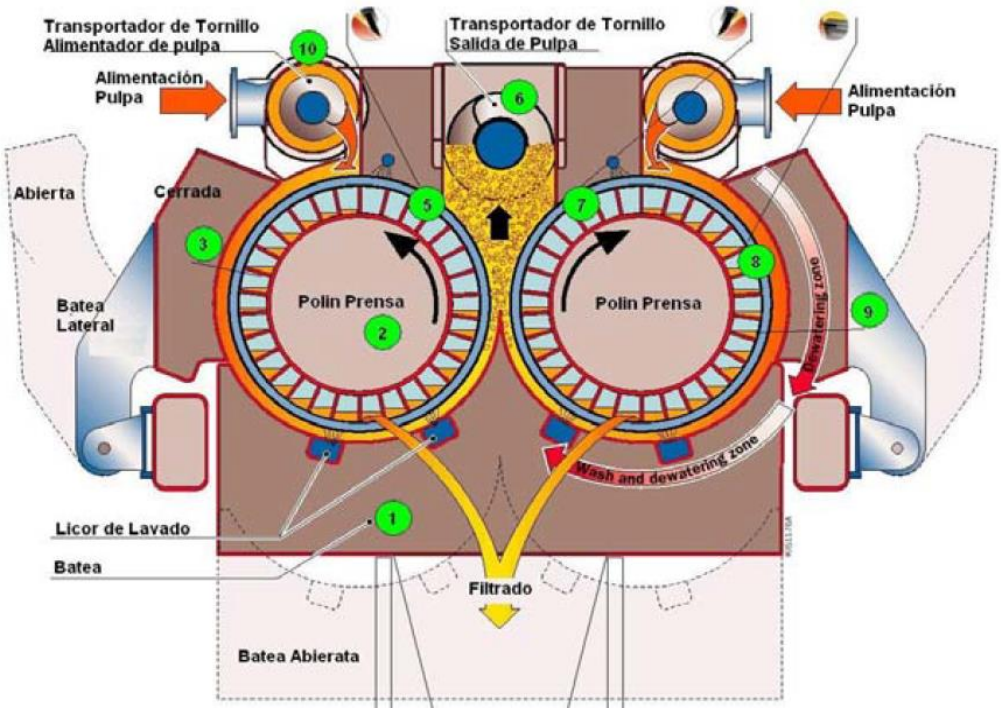


Figura 2.4: Corte transversal de una prensa de desplazamiento [5].

La pulpa ingresa por la parte superior de la prensa y es dosificada por los Tornillos de Distribución (10) asegurando un flujo uniforme para ambos Rodillos o Polines perforados (2). Entre el Rodillo y el Flap (o Batea Lateral) ocurre una primera etapa de desagüe, donde el filtrado pasa por las perforaciones del Rodillo, llegando a una consistencia estimada del 15%. Luego viene el lavado por desplazamiento donde se inyecta Licor de Lavado a través dos corridas de boquillas ubicadas en el fondo de la Batea (1). La pulpa forzada por el movimiento de los rodillos llega a la parte central donde es exprimida por segunda vez entre los Rodillos y la Batea, alcanzando una consistencia aproximada del 35%. Finalmente, la pulpa sale por el Tornillo Desmenuzador (6) ubicado en la parte central [5].

Existen Raspadores (5) y Duchas (7) que adicionan filtrado a alta presión para limpiar los Rodillos. El manto de los Rodillos es perforado y en su interior tienen una serie de Pletinas (3) soldadas y dispuestas en forma radial, y una Camisa (9) contraída sobre estas con canales de drenaje.

Figura 2.5: Motor hidráulico (de color naranja) y rodillo en el fondo.

Figura 2.6: Batea de una prensa de desplazamiento.

Los Rodillos son accionados por Motores Hidráulicos (Figura 2.5) controlados por el Sistema Hidráulico de la Figura 2.7.

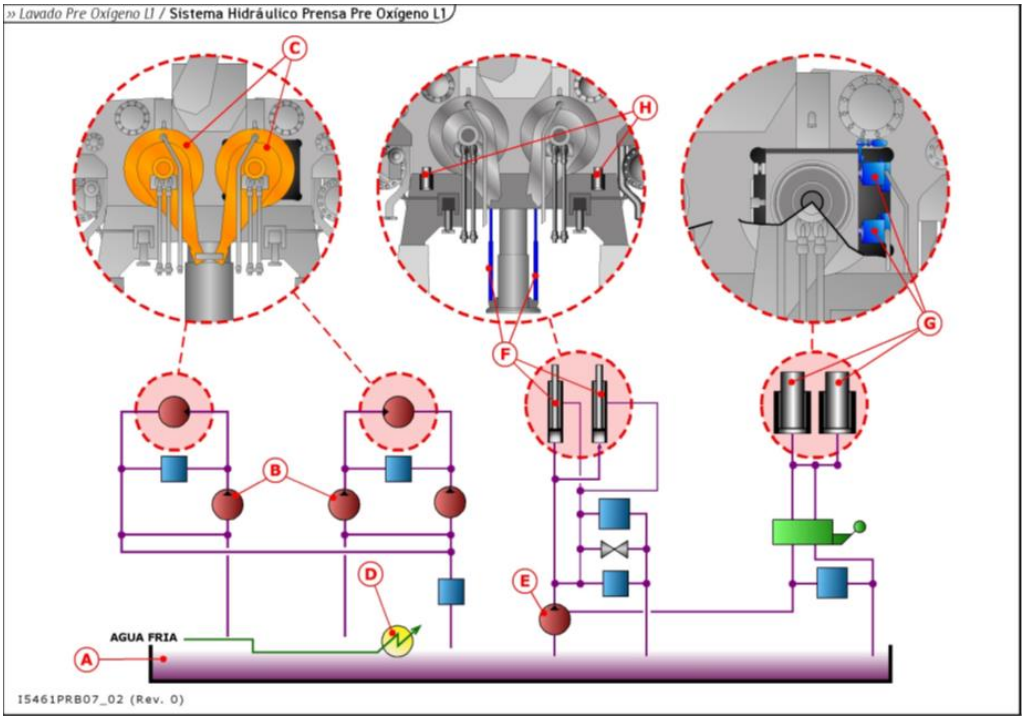


Figura 2.7: Sistema hidráulico de una prensa de desplazamiento [6].

Las Bombas Principales (B) envían aceite desde el Estanque (A) hacia los Motores Hidráulicos (C). El flujo de aceite hace girar los Motores y en consecuencia los Rodillos de la Prensa. Una Bomba Hidráulica (E) envía aceite hacia los Pistones Hidráulicos de Batea (G) y hacia los Pistones Hidráulicos de los Flap (H), permitiendo ajustar la separación Rodillos-Batea y Rodillos-Flap respectivamente. Un Intercambiador de Calor (D) enfría el aceite [6].

Es importante señalar que uno de los Rodillos es móvil mientras que el otro es fijo en el plano horizontal, y la separación entre ellos influye directamente en la consistencia de descarga de la pulpa. Los Pistones Hidráulicos de Carga (F) están encargados de mantener la separación de los rodillos dentro de cierto rango. Cuando el esfuerzo producido por el flujo de pulpa es excesivo, una válvula de sobrepresión recircula el aceite, permitiendo la separación de los rodillos y detención de la prensa.

Otro aspecto importante a considerar es el siguiente. La presión de aceite en la línea de los motores hidráulicos es una señal indirecta del esfuerzo requerido para hacer girar los rodillos, en otras palabras, el torque. En condiciones normales el Sistema de Control aumenta

el flujo de aceite en los motores hidráulicos para aumentar la velocidad de los rodillos, resultando en un aumento del torque impuesto por la carga, esto es, la pulpa que está siendo lavada.

El torque que mide el Sistema de Control depende de factores como la drenabilidad de la pulpa, consistencia de entrada y el flujo de alimentación. Variaciones en el torque influyen directamente en la consistencia de descarga de la pulpa.

Figura 2.8: Unidad hidráulica encargada de accionar los rodillos de una prensa.

2.4. Industria 4.0

A lo largo de la historia el desarrollo y aplicación de nuevas tecnologías ha impactado enormemente a la sociedad particularmente en los ámbitos económico e industrial con un aumento exponencial de los niveles de producción. El primer gran hito, conocido como la primera revolución industrial, está marcada por la mecanización y la utilización del agua y vapor como fuentes de energía para el accionamiento de maquinaria industrial, destacando la máquina a vapor. La segunda revolución se define por la utilización de la energía eléctrica en los procesos industriales, así como la línea de ensamblado que permitió la producción en masa. La tercera revolución es más reciente y viene de la mano de la informática, con el desarrollo de computadores y el internet que permitieron la automatización de tareas repetitivas gracias a la programación de controladores y robots, así como también una rápida expansión del área de servicios [7].

El concepto de Industria 4.0 hace referencia a la cuarta revolución industrial que se avecina con la introducción de tecnologías del internet en la industria. El término fue usado por primera vez en 2011 por líderes de la industria principalmente de Alemania. Desde ese momento se ha tratado de clarificar cuales son los límites, beneficios y tecnologías habilitadoras de la Industria 4.0. Algunos principios fundamentales de la cuarta revolución industrial ya han sido introducidos como el uso extensivo del internet, no solo usándolo para conectar dispositivos, sensores y equipos sino también creando nuevas funciones en los productos que usan el Internet como fuente de información, la flexibilidad de la producción, relacionado con la manufactura aditiva y trazabilidad de productos, y la virtualización de los procesos con los sistemas ciberfísicos [7].

El surgimiento de nuevas fábricas inteligentes y su gestión online de la producción, la masificación de los sistemas de información, la digitalización y la abundancia de sensores y dispositivos en las industrias son indicadores de que la cuarta revolución industrial ya está en marcha. Entre algunas tecnologías habilitadoras de la Industria 4.0 están: Internet de las Cosas (IoT), Sistemas Ciberfísicos (CPS), Big Data, Data Analytics, Data Mining, Cloud Computing, Ciberseguridad, Inteligencia Artificial (AI) y Gemelo Digital (DT).

Para tener una mejor idea de qué constituye la industria 4.0 a continuación se describen algunas tecnologías habilitadoras [8]:

Internet de las cosas

Corresponde a la interconexión digital entre dispositivos y equipos de manera que puedan comunicarse e interactuar entre sí en tiempo real, recopilando, enviando y recibiendo información. Por su parte, el Intenet Industrial de las Cosas (IIoT) se enfoca en aplicaciones industriales como transporte, manufactura, energía y agricultura, donde es posible

implementar procesos autónomos que recopilen información desde sensores que permitan el monitoreo de equipos de manera remota.

Sistemas Ciberfísicos

Corresponden a la integración de los procesos físicos con lo digital por medio de dispositivos con capacidades computacionales, de almacenamiento y comunicación, que son capaces de controlar y monitorear procesos físicos. El humano puede intervenir por medio de interfaces de comunicación

Big Data, Data Analytics y Data mining

Cuando el flujo de datos en términos de volumen, velocidad y variedad no es manejable por sistemas de base de datos tradicionales se habla de Big Data. Estos conceptos hacen referencia a la gestión e interpretación del gran volumen de datos generado por tecnologías IoT y Sistemas Ciberfísicos, con el fin de mejorar la productividad, eficiencia y calidad en los procesos y servicios como también mejorar la toma de decisiones en cuanto a desarrollo de productos.

Cloud Computing

Hace referencia a la utilización de recursos computacionales, especialmente de almacenamiento de datos y poder computacional en el *Internet*, sin disponer de infraestructura física.

Ciberseguridad

La introducción de estas tecnologías 4.0 en entornos industriales requiere de sistemas de información y comunicación que aseguren la protección, privacidad y seguridad de las empresas frente a amenazas como ciber-ataques.

Inteligencia Artificial

Sistemas capaces de procesar y aprender de la información de manera autónoma con el fin de realizar tareas complejas emulando el funcionamiento de la inteligencia humana.

Gemelo Digital

Modelos virtuales de servicios, activos, sistemas o procesos que permiten su optimización, monitoreo y/o control sin necesidad de acceder a su contra-parte física.

2.4.1. Inteligencia Artificial

La Inteligencia Artificial, o AI por sus siglas en inglés, es una rama de la ciencia computacional que corresponde al estudio y desarrollo de máquinas inteligentes y softwares que pueden razonar, aprender, recopilar conocimiento, comunicar, manipular y percibir objetos [9]. La inteligencia artificial tiene una serie de aplicaciones tales como: Visión Computacional aplicada en el reconocimiento de imágenes y extracción de información a partir de entradas visuales; Servicio al Cliente implementado por medio de agentes virtuales; Reconocimiento de Discurso mediante el procesamiento del lenguaje natural para transcribir lo percibido por voz a un formato escrito; Motores de búsqueda que descubren tendencias en datos de consumo y recomiendan información relevante al consumidor, entre muchas otras aplicaciones.

2.4.2. Machine Learning

El Aprendizaje Automático o *Machine Learning* es una rama de la Inteligencia Artificial que corresponde al desarrollo de algoritmos y modelos estadísticos para realizar tareas específicas sin el uso explícito de instrucciones para su programación. Estos algoritmos son capaces de aprender de la experiencia, en otras palabras, pueden reconocer patrones y generalizar comportamientos a partir de un conjunto de datos, y en base a esto tomar decisiones o hacer predicciones con precisión.

Una rama del Aprendizaje Automático es el Aprendizaje Profundo o *Deep Learning*, que corresponde al aprendizaje a través de representaciones de los datos de entrada en una estructura de capas sucesivas, de manera que cada capa es una transformación de la información obtenida en la capa anterior [10]. Las Redes Neuronales son los modelos más representativos del Aprendizaje Profundo.

Según el método de aprendizaje, los modelos y algoritmos de *Machine Learning* son clasificados en 4 grupos [11].

Supervisado

En este tipo de aprendizaje, los datos de entrenamiento están etiquetados, en otras palabras, el valor de salida asociado a cada valor de entrada es conocido de antemano. A

través de la optimización iterativa de una función objetivo el algoritmo aprende, es decir, encuentra una función que dada las variables de entrada (*Input*), asigna una etiqueta o valor de salida (*Output*) adecuado. Dentro de esta categoría se encuentran los métodos de clasificación, donde la salida corresponde a clases discretas, y los métodos de regresión, donde la salida es un valor continuo.

No-Supervisado

En el caso de métodos no supervisados, los datos de entrenamiento no están etiquetados o no presentan una estructura conocida, es decir, no se conoce el valor de salida asociado a cada entrada. Este tipo de método permite explorar la estructura de estos datos para extraer información significativa sin la ayuda de variables de resultado conocidas. Los métodos de agrupamiento y de reducción de dimensionalidad son parte de este tipo de aprendizaje.

Semi-Supervisado

Como su nombre lo indica este tipo de aprendizaje combina aspectos de los métodos supervisados y no supervisados. Un ejemplo particular de este tipo de aprendizaje son los métodos de detección de novedades donde los datos de entrenamiento están etiquetados mientras que los datos de prueba no lo están.

Refuerzo

El objetivo en este tipo de aprendizaje es desarrollar un sistema o agente que mejore el rendimiento del algoritmo basado en interacciones con el entorno. El aprendizaje se realiza mediante un proceso de retroalimentación donde el agente entrega una señal de recompensa cuando el modelo tiene un buen desempeño, o una señal de castigo en caso contrario, es decir, el aprendizaje está basado en ensayo y error.

Los algoritmos y modelos de Aprendizaje Automático no solamente se clasifican según el tipo de aprendizaje, también pueden ordenarse de acuerdo con su aplicación, como se describe a continuación [11]

Clasificación

Como se mencionó, corresponden a modelos de aprendizaje supervisado, y su objetivo es identificar a qué clase pertenece una nueva observación a partir de un conjunto de datos cuya categoría es conocida, en este caso las etiquetas de clase son discretas. Entre

los algoritmos de clasificación más utilizados se encuentran: K-Nearest Neighbors, Decision Forest, Random Forest, Support Vector Machine. Las Redes Neuronales también pueden utilizarse como modelos de clasificación.

Regresión

Estos modelos permiten predecir valores continuos en base a una o más variables predictoras o explicativas mediante un ajuste a los datos de entrada. El método de regresión más básico es el de la regresión lineal, donde el conjunto de datos se ajusta mediante una combinación lineal de éstos, buscando minimizar el error cuadrático medio. Las Redes Neuronales también pueden usarse como modelos de regresión.

Agrupamiento

Esta técnica exploratoria no supervisada permite organizar los datos en subgrupos o *clusters* sin conocer las etiquetas o clases. Los datos son agrupados en base a criterios de distancia o similitud, de esta manera estos métodos permiten estructurar la información y derivar relaciones significativas de los datos. Algunos de los algoritmos de agrupamiento más conocidos son: K-Means, Mean Shift Clustering, DBSCAN, Gaussian Mixture Models y Agglomerative Hierarchical Clustering.

Reducción de Dimensionalidad

Los datos muchas veces poseen una alta dimensionalidad, lo cual resulta ser condicionante en términos de almacenamiento y capacidad computacional. Este tipo de algoritmos se utiliza por lo general en etapas de procesamiento de datos y permite reducir la cantidad de los datos conservando gran parte de la información y eliminando redundancia en los datos. Lo anterior permite disminuir los tiempos de entrenamiento y mejorar el desempeño de los algoritmos. La reducción de dimensionalidad también puede ser útil para la visualización de los datos. Entre los métodos de reducción más usados se tienen: Principal Component Analysis, Kernel PCA, Factor Analysis, Linear Discriminant Analysis, Multidimensional Scaling, ISOMAP y Local Linear Embedding.

Detección de Anomalías/Novedades

La detección de anomalías consiste en identificar elementos que parecen desviarse notoriamente de los otros miembros de un conjunto de datos [12]. Dependiendo de las características de los datos de entrenamiento estos métodos pueden clasificarse en dos categorías: Detección de anomalías, donde los datos de entrenamiento contienen datos

anómalos y se debe estimar la proporción de éstos con respecto a los datos normales; Detección de novedades, donde los datos de entrenamiento son exclusivamente datos etiquetados como "sanos" mientras los datos de prueba no están etiquetados. Algunos algoritmos de detección conocidos son: Robust Covariance, One Class Support Vector Machine, Local Outlier Factor y Isolation Forest. Las Redes Neuronales también pueden utilizarse para la detección de anomalías.

2.4.3. Modelos utilizados

A continuación, se describen en detalle los algoritmos y modelos utilizados en ambos casos de estudio.

Algoritmos de agrupamiento

i) K-Means

Este algoritmo agrupa los datos en base a la distancia entre puntos, tratando de separarlos en *k* grupos de igual varianza. El número de grupos *k* debe ser especificado [13].

El algoritmo opera mediante los siguientes pasos:

1. Elegir aleatoriamente los *"k"* centroides iniciales
2. Cada muestra es asignada a su centroide más cercano
3. Re-calcular centroides como el promedio de las muestras de cada *cluster*

Se iteran los últimos 2 pasos hasta que el algoritmo converge, en otras palabras, cuando la posición de los centroides no varía significativamente.

El algoritmo busca centroides que minimicen la inercia o suma de cuadrados dentro de cada *cluster*. La inercia está definida por la Ecuación 2.1.

$$\sum_{i=0}^{n}\left(\|x_i - \mu_j\|^2\right) \qquad (2.1)$$

El algoritmo divide un set de *n* muestras x_i en *k clusters*. Cada *cluster* está descrito por su centroide μ_j.

Existen algunas limitaciones del algoritmo. En el cálculo de la inercia se asume que los *clusters* son convexos e isotrópicos, por lo que su rendimiento no es muy bueno para grupos de formas irregulares

La inercia no es una métrica normalizada, por tanto, cuando se trabaja con datos de múltiples dimensiones, las distancias euclidianas tienden a inflarse. Esto puede aliviarse utilizando métodos de reducción de dimensionalidad como PCA (*Principal Component Analysis*).

ii) Mean Shift

Algoritmo de agrupamiento basado en centroides que trabaja de la siguiente forma. En cada iteración se calculan nuevos centroides que son el promedio de los puntos de una determinada región, esta región tiene un tamaño *Bandwidth*, que es el único parámetro ajustable [13]. Los centroides se actualizan según la Ecuación 2.2.

$$x_i^{t+1} = m(x_i^t) \qquad (2.2)$$

Donde *m* es el vector *mean shift* calculado para cada centroide. El vector apunta en la dirección de máxima densidad y está definido por la Ecuación 2.3.

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)} \qquad (2.3)$$

Donde $N(x_i)$ es la vecindad de cada muestra dada una distancia alrededor de x_i.

A diferencia de K-Means el algoritmo Mean Shift encuentra por sí solo el número de *clusters*, aunque se debe mencionar que la selección del parámetro *Bandwidth* no es trivial.

iii) DBSCAN

Algoritmo basado en densidad, de modo que identifica los *clusters* como áreas de alta densidad de puntos separadas por áreas de baja densidad. Esto es una ventaja en el sentido que el *cluster* puede tener cualquier forma, a diferencia de K-Means por ejemplo que asume una forma convexa. [13]

Para entender el funcionamiento de este algoritmo se definen algunos conceptos. Una muestra central (*Core sample*) es cualquier muestra tal que existen al menos una cantidad *min_samples* de muestras a una distancia *epsilon*, llamados vecinos (*neighbors*) de la muestra central. Cada muestra central está por definición ubicada en un área densa, por tanto, un *cluster* se construye de forma recursiva tomando una muestra central, encontrando sus vecinos que también son muestras centrales, y así sucesivamente. El *cluster* también está formado por muestras no-centrales, pero que pertenecen al conjunto de vecinos de una muestra central. Por último, están las muestras aisladas (*outliers)* que se definen como muestras no-centrales que no están en el conjunto de vecinos de una muestra central.

Los parámetros ajustables del algoritmo son: *min_samples*, que controla principalmente la tolerancia al ruido; *epsilon*, que cuando es muy pequeño no se generan *clusters* y todos los puntos son etiquetados como ruido (-1), mientras que, si es muy grande, todos los puntos son agrupados en un solo *cluster*. Cabe destacar que ambos parámetros están vinculados de cierta forma, fijar un *epsilon* mayor tiene un efecto similar a establecer un *min_samples* menor.

Existen algunas recomendaciones y métodos para estimar estos parámetros. Para estimar el parámetro *min_samples* se recomienda [14]:

- *min_samples = 2*dimensiones* (parámetros)

Para estimar el parámetro *epsilon* existen métodos cómo el mostrado en [15], donde para cada punto se calcula la distancia promedio de sus *k* vecinos. Luego se grafican las distancias promedio en orden ascendente en función de los puntos. El valor óptimo de *epsilon* corresponde al punto de máxima curvatura.

iv) Gaussian Mixture Models

Modelos similares a K-Means pero de mayor flexibilidad, puesto que asumen una distribución Gaussiana de los datos, de modo que se tienen dos parámetros para describir los *clusters*, la media y la desviación estándar [13].

Para encontrar los parámetros de cada *cluster* el modelo usa un algoritmo de optimización llamado *Expectation Maximization*. Sin embargo, el modelo requiere el ajuste de un solo parámetro: *n_components*. Los pasos iterativos para la optimización de los parámetros de distribución Gaussiana se describen a continuación.

1. Iniciar con parámetros aleatorios para cada *cluster*
2. Calcular probabilidad de cada punto de pertencer a un *cluster* en particular
3. Re-calcular parámetros para distribuciones Gaussianas.

Se repiten los pasos 2 y 3 hasta la convergencia, es decir, hasta maximizar las probabilidades de cada muestra de pertenecer a un *cluster*.

v) Agglomerative Hierarchical Clustering

El agrupamiento jerárquico (*Hierarchical clustering*) es una familia de algoritmos que construyen *clusters* mediante la combinación o separación sucesiva de éstos [13]. En el caso de los *Agglomerative clustering*, el proceso de formación de *clusters* es el siguiente.

En la primera iteración cada muestra es su propio *cluster*, en la siguiente iteración los *clusters* más cercanos se unen y forman uno solo, y así de forma sucesiva hasta llegar al número de *clusters* indicado por el parámetro ajustable *n_clusters.*

Existen diferentes criterios de vinculación para la unión de los *clusters*, entre ellos se tienen.

- Vinculación completa: Distancia de las muestras más alejadas de cada *cluster*.
- Vinculación promedio: Distancia promedio de las muestras de cada *cluster*.
- Vinculación única: Distancia de las muestras más cercanas de cada *cluster*.

vi) HDBSCAN

Es un algoritmo de agrupamiento desarrollado por Campello, Moulavi y Sander [16]. Corresponde a una extensión de DBSCAN al convertirlo en un algoritmo de agrupamiento jerárquico.

HDBSCAN realiza iteraciones del algoritmo DBSCAN sobre diferentes valores de *epsilon* para luego integrar el resultado y encontrar un agrupamiento que entregue la mejor estabilidad sobre *epsilon.* Esto permite encontrar *clusters* de diferentes densidades a diferencia de DBSCAN. [17]

Existen varios parámetros ajustables, sin embargo aquellos que tienen una mayor influencia en la formación de *clusters* son 3: El parámetro *min_cluster_size* corresponde a la cantidad mínima de muestras que debe tener un grupo para ser considerado *cluster*; El parámetro *min_samples* al igual que en DBSCAN es la cantidad mínima de vecinos que debe tener una muestra para ser considerada muestra central (*core sample)*; El parámetro *cluster_selection_epsilon* que funciona como un límite de distancia donde los *clusters* que están a una distancia menor al parámetro se unen en un solo *cluster*.

HDBSCAN tiene la capacidad de asignar nuevas muestras a un conjunto de *clusters* previamente existentes haciendo uso de la función *approximate_predict()*. Sin embargo, en la actual versión de HDBSCAN los efectos del parámetro *cluster_selection_epsilon* no están totalmente integrados y algunas funciones, entre ellas *approximate_predict()*, no toman el parámetro en consideración.

Para resolver lo anterior se añadió un nuevo módulo "*flat"* al algoritmo HDBSCAN [18]. Los parámetros ajustables en este módulo son principalmente 2: El parámetro *min_cluster_size* que es básicamente el mismo anteriormente descrito; Y el parámetro *n_clusters* donde se fija el número de *clusters*.

Algoritmos de detección de Anomalías/Novedades

i) Elliptic Envelope

Este método asume que los datos de entrada tienen una distribución Gaussiana, y cualquier muestra que estén lo suficientemente alejadas de la forma ajustada, será considerado anómalo.

El algoritmo ajusta una estimación de covarianza robusta a los datos, esto quiere decir, sin la influencia de datos anómalos. En la práctica se ajusta una elipse a los datos centrales ignorando los datos aislados o anómalos [19].

Para tener una medida del grado de anomalía de cada dato se usa la distancia de Mahalanobis definida por la Ecuación 2.4.

$$Md(x, \mu, \Sigma) = \sqrt{(x-\mu)\Sigma^{-1}(x-\mu)^T} \qquad (2.4)$$

Donde μ es la media de los datos y Σ la matriz de covarianzas. La estrategia del algoritmo se ilustra en la Figura 2.9.

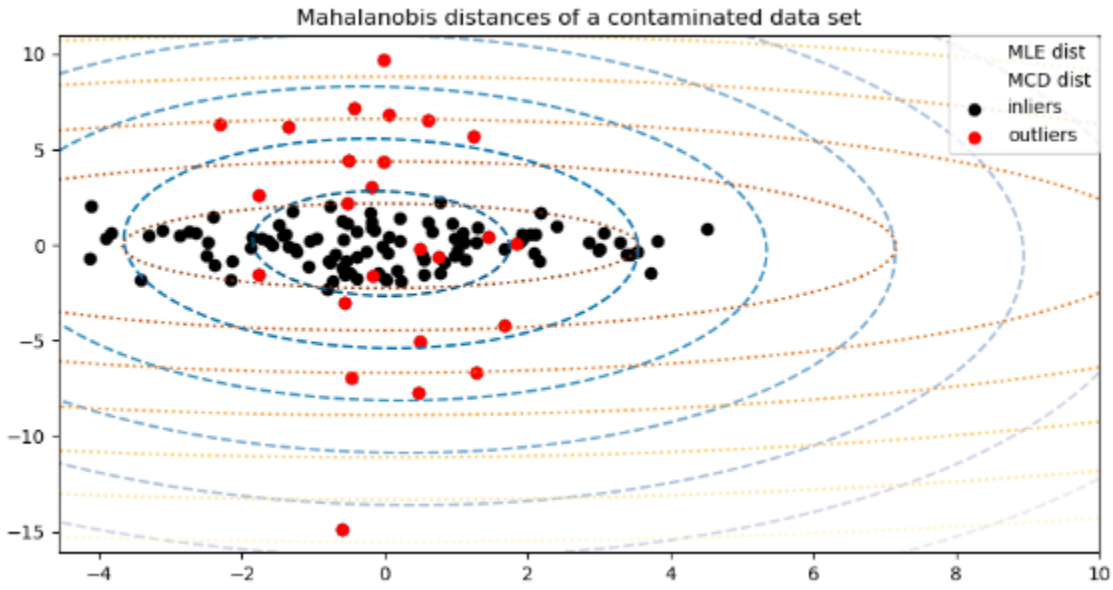


Figura 2.9: Estrategia de clasificación del algoritmo Elliptic Envelope [20].

ii) Isolation Forest

Una forma eficiente de identificar elementos anómalos en conjuntos de datos de altas dimensiones es mediante el uso de algoritmos *Random Forest.* Este algoritmo en particular aísla muestras seleccionando aleatoriamente un parámetro y luego seleccionando aleatoriamente un valor de división entre los valores máximos y mínimos de dicho parámetro, particionando los datos en el valor de división [19].

Esta partición de los datos se puede hacer de forma iterativa hasta aislar completamente una muestra. De esta forma, la cantidad de particiones que se debe hacer para aislar una muestra se toma como una medida de anormalidad.

El proceso de partición puede ser representado por una estructura de árbol, donde el número de divisiones requeridos para aislar una muestra es equivalente al largo del trayecto desde el nodo-raíz a los nodos-hojas. Por tanto, cuando un “bosque” de “árboles” aleatorios producen trayectorias cortas para una muestra, hay una alta probabilidad de que la muestra sea anómala. La estrategia del algoritmo se ilustra en la Figura 2.10.

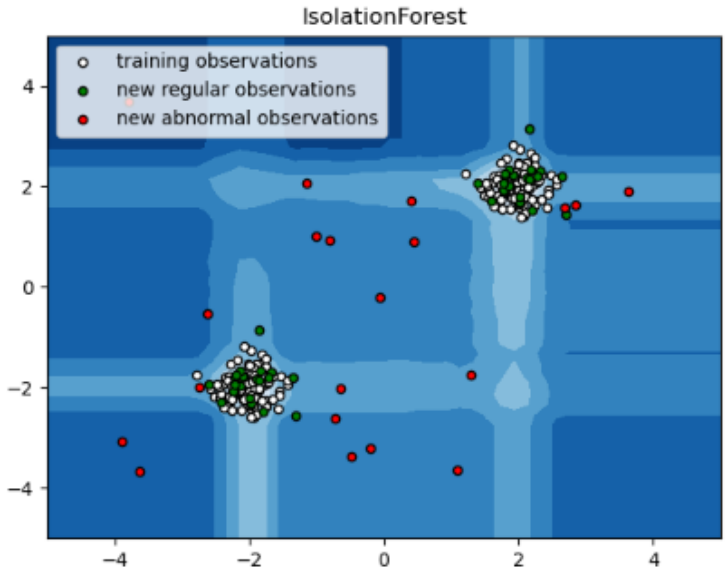


Figura 2.10: Estrategia de clasificación del algoritmo Isolation Forest [20].

iii) Local Outlier Factor

Este algoritmo calcula un puntaje de anormalidad para cada muestra, el cual mide la densidad local de la muestra con respecto a sus vecinos, de manera que las muestras con densidades notablemente inferiores a las de sus vecinos son consideradas anómalas [19].

La densidad local es obtenida de los *k* vecinos más cercanos, mientras que el puntaje LOF de una muestra es igual al ratio entre, la densidad local promedio de los *k* vecinos más cercanos y la densidad local de la muestra. Por lo tanto, para datos normales se espera que la densidad local sea similar a la de sus vecinos (puntaje LOF ~ 1), mientras que para datos anormales la densidad local de la muestra sería mucho menor (puntaje LOF >> 1). La Figura 2.11 muestra la estrategia del algoritmo.

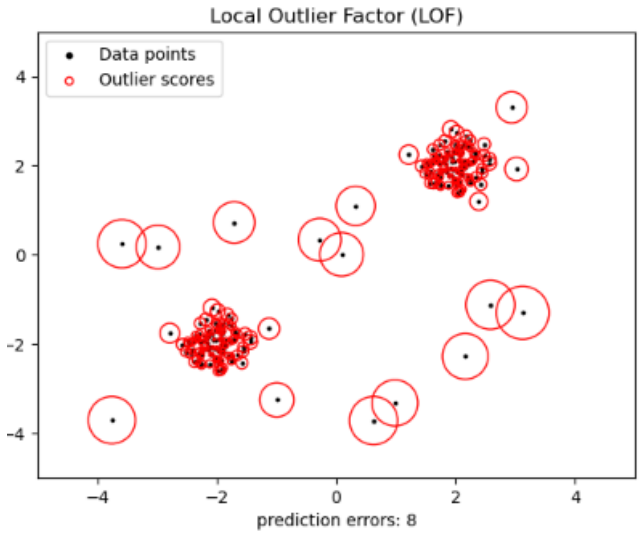


Figura 2.11: Estrategia de clasificación del algoritmo Local Outlier Factor.

iv) One Class Support Vector Machine

One Class Support Vector Machines es un algoritmo no supervisado usado para problemas de clasificación binarios incluyendo detección de anomalías. Este algoritmo construye una hiper-esfera de radio mínimo que se ajusta a las muestras de una única clase, con la idea de minimizar su radio *r* como muestra la Figura 2.12, considerando como anómalos aquellas muestras que quedan fuera de la hiper-esfera [20].

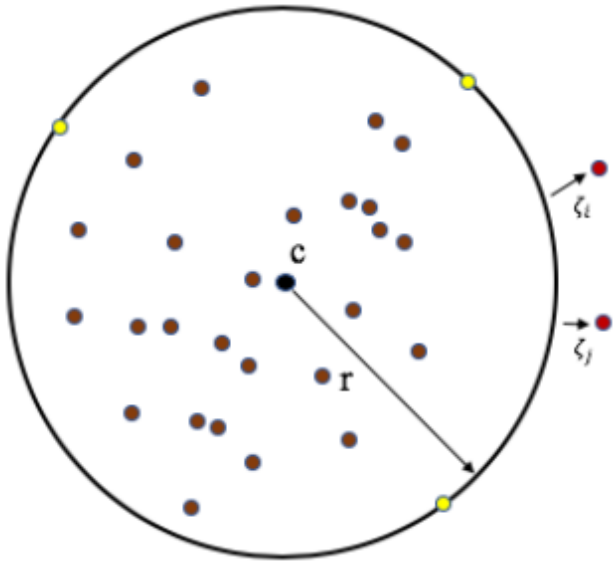


Figura 2.12: Estrategia de clasificación del algoritmo One Class Support Vector Machines [20].

OC-SVM es conocido por ser sensible a datos anómalos, de modo que su desempeño no es muy bueno para la detección de anomalías. El método puede ser usado, pero requiere un ajuste fino del hiper-parámetro *nu*, tanto para el manejo de los datos anómalos como para la prevención del sobreajuste.

Redes Neuronales

Las redes neuronales (*Neural Networks*) corresponden a modelos de aprendizaje supervisado. Su unidad base es el perceptrón y tiene una estructura similar a las neuronas biológicas como se muestra a la izquierda de la Figura 2.13.

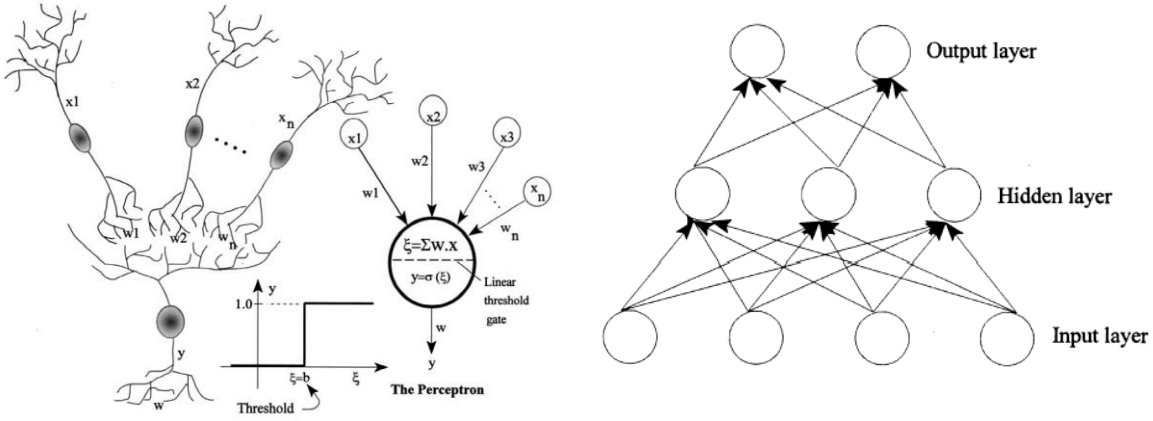


Figura 2.13: Izquierda: Esquema de Perceptrón; Derecha: Esquema de red multicapa [22].

El perceptrón posee 3 componente principales: Las entradas (x_i); La neurona, que primero realiza una suma ponderada entre las entradas (x_i) y sus respectivos pesos (w_i), para luego aplicar una función de activación (σ) siempre y cuando la suma ponderada sea mayor al sesgo (b); Por último, la salida (y) corresponde al resultado de las operaciones anteriores [21]. Cabe mencionar que la función de activación es de carácter no-lineal, lo que permite disociar una capa de la otra y de esta forma cada capa entrega una nueva representación de la información extraída por las capas anteriores. Si la activación fuera lineal, la salida de cada capa sería solamente una transformación lineal de las anteriores [22]. Las funciones de activación más comunes son la Sigmoide, Tangente hiperbólica, *Rectified Linear Unit (relu)*, y Softmax.

Una red neuronal está formada por capas como muestra la derecha de la Figura 2.13: Una capa de entrada, que tiene tantas neuronas como variables de entrada; Una o varias capas ocultas; Una capa de salida que puede tener una o varias neuronas dependiendo del problema que se quiera resolver, para problemas de clasificación binaria la capa de salida tiene una

sola neurona con activación Sigmoide. Esta estructura multi-capa es también conocida como *Multi-Layer Perceptron.*

El aprendizaje de una red neuronal consiste en encontrar una combinación de sus parámetros (pesos y sesgos) tal que a cada entrada se le asigne su correspondiente salida. Para encontrar esta combinación de parámetros se comienza inicializando con valores aleatorios de estos parámetros. Una función de pérdida calcula el error entre el valor de salida y el valor esperado, en un problema de clasificación binaria la función de pérdida utilizada es la *binary_crossentropy*. Este error es propagado capa por capa y los pesos van siendo actualizados en base al error. Este algoritmo es conocido como *Backpropagation* y es la forma en que la red "aprende". Los pesos son actualizados de forma iterativa hasta la convergencia.

Redes Neuronales Recurrentes

Las redes neuronales por lo general procesan la información de forma independiente, de modo que no importa el orden en que son ingresados los datos. Sin embargo, en ciertos problemas los datos siguen una secuencia, como el caso de las series de tiempo, donde existe un orden temporal.

Las redes neuronales recurrentes (*RNN*) procesan secuencias de la información iterando sobre cada elemento de la secuencia y guardando la información más relevante, este proceso es detallado en la Figura 2.14. Se observa que en cada iteración aparte de la salida (*output)*, existe un estado (*State)*, el cual conserva información de las iteraciones anteriores y toma parte en el cálculo de cada valor de salida [10].

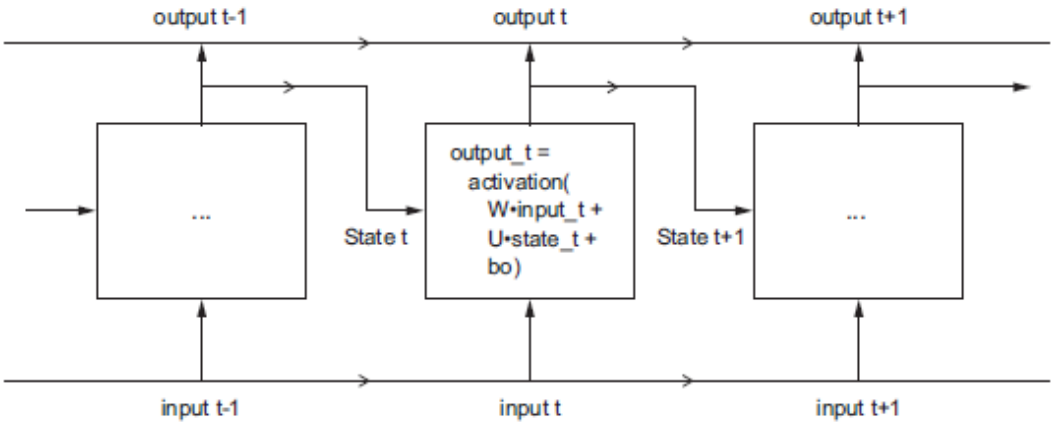


Figura 2.14: Esquema de una Red Neuronal Recurrente [10].

Un tipo de capa de las *RNN* es la *Long-Short Term Memory (LSTM)*, la cual resuelve algunos problemas presentes en las capas recurrentes simples (*vanishing gradient problem*), para ello guarda la información pasada de una forma un tanto diferente, y es explicada a continuación en 4 pasos, usando como referencia la Figura 2.15 [22].

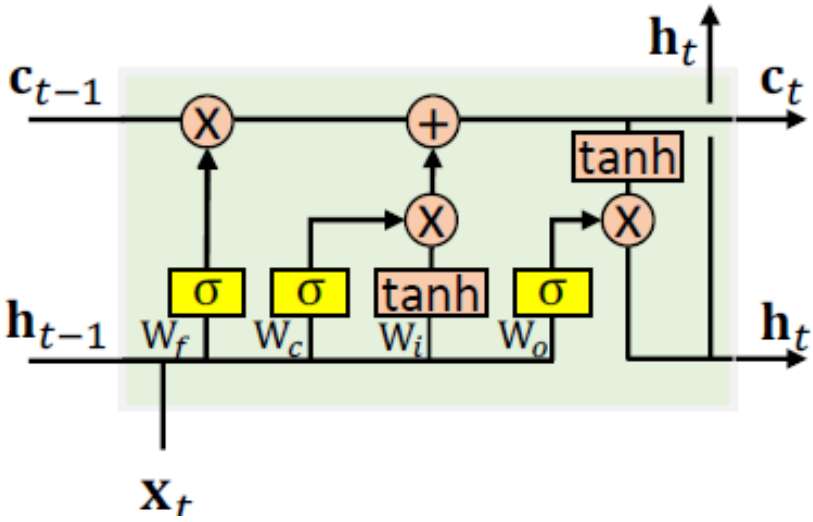


Figura 2.15: Esquema de una neurona de capa LSTM [22].

1. En primer lugar, se procesa la información de la entrada (x_t) junto con la información de la iteración anterior (h_{t-1}) utilizando las Ecuaciones 2.5 y 2.6.

$$\bar{c_t} = tanh(w_c h_{t-1} + w_c x_t + b_c) \quad (2.5)$$

$$i_t = \sigma(w_i h_{t-1} + w_i x_t + b_i) \quad (2.6)$$

   La primera ecuación encuentra información relevante en (x_t), mientras que la segunda, conocida como *Input gate*, mide la relevancia de la información de entrada.

2. Frente a la nueva información procesada es posible que parte de la información pasada ya no sea relevante y por tanto debe ser olvidada. Este procesamiento se realiza con la Ecuación 2.7 conocida como *Forget gate.*

$$f_t = \sigma(w_f h_{t-1} + w_f x_t + b_f) \quad (2.7)$$

3. La información que será conservada y que pasa a la siguiente iteración, viene dada por la Ecuación 2.8 conocida como *Cell state*, y se puede asociar a la memoria a largo plazo.

$$c_t = f_t c_{t-1} + i_t \bar{c}_t \qquad (2.8)$$

El primer componente de la ecuación determina si se conserva la información anterior, mientras que el segundo componente contiene la información relevante encontrada en la iteración *t*.

4. Finalmente se calcula la salida (h_t) de la iteración *t* mediante la Ecuación 2.9.

$$h_t = o_t tanh(c_t) \qquad (2.9)$$

Donde (o_t) es el *Output gate* y se calcula a partir de la Ecuación 2.10.

$$o_t = \sigma(w_o h_{t-1} + w_o x_t + b_o) \qquad (2.10)$$

El término (h_t) se asocia a la memoria a corto plazo, y corresponde al valor de salida de la capa *LSTM*. En algunos casos se requiere que el valor de salida de la red sea una secuencia, en cuyo caso la red recibe el nombre de *unrestricted*, mientras que, si sólo se devuelve el último *output* calculado, la red se define como *restricted.*

Las redes *LSTM* también pueden ser *Stateless*, cuando se quiere procesar secuencias de información de forma independiente, o *Stateful*, cuando se procesan sub-secuencias de una secuencia mayor.

Las capas *LSTM* procesan la información en el orden en que es entregada. El orden de procesamiento puede ser invertido usando el argumento *Backwards*, y de esta forma extraer nueva información. También existen las capas *Bidirectional*, que procesan la información en ambos sentidos.

Redes Neuronales Convolucionales

Como su nombre lo indica, las redes neuronales convolucionales (*CNN*) utilizan la operación matemática de la convolución para procesar información. La convolución permite encontrar patrones en los datos de entrada mediante la utilización de filtros (funciones kernel), y es muy utilizado en aplicaciones de reconocimiento de imágenes [22].

Una *CNN* típicamente está formada por una capa convolucional y una capa *Pooling*, como se muestra en la Figura 2.16

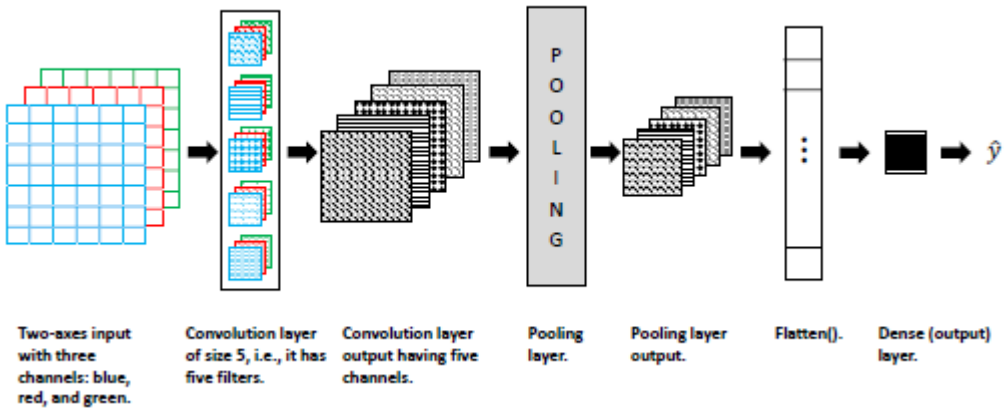


Figura 2.16: Arquitectura de un Red Neuronal Convolucional típica [22].

El *input* comúnmente tiene una estructura de grilla (2D) y puede o no tener una tercera dimensión, que en el caso del ejemplo de la figura corresponde a los canales RGB de una imagen.

La capa convolucional contiene 5 unidades, cada una de ellas contiene un filtro diferente. Al realizar la operación de convolución entre el *input* y el kernel la información de los 3 canales se comprime en un solo canal. Se obtiene un *output* de 5 canales por cada filtro aplicado.

La capa *Pooling* es aplicada para extraer la información más representativa reduciendo la cantidad de datos y haciendo más eficiente el procesamiento de la información.

Una capa *Flatten* es incorporada para vectorizar la matriz resultante de la capa anterior y entregar la información a la capa de salida.

Una forma de procesar series temporales multivariables en una red convolucional es usando capas convolucionales *Conv1D*, donde los datos deben tener una estructura como la

mostrada en la Figura 2.17, donde cada canal corresponde a una variable de entrada o parámetro.

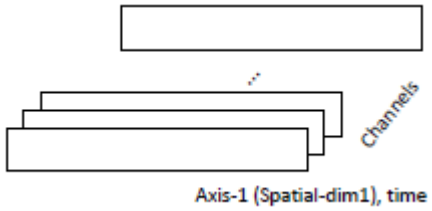


Figura 2.17: Estructura de series temporales multivariables para ser procesadas por capas Conv1D. [22]

Existen variantes de la convolución que pueden ayudar en el procesamiento de la información. Cada operación de convolución reduce las dimensiones de los datos de entrada, restringiendo la cantidad de capas convolucionales que se pueden aplicar. La variante *Padding* permite que el tamaño del *output* sea igual al *input* mediante la incorporación de columnas y/o filas de ceros en la periferia del *input.* Otra variante es el *Stride,* que básicamente define la cantidad de pasos en que es aplicado el filtro. La variante *Dilation* por su parte, expande el filtro agregando espacios "vacíos" entre sus elementos.

Otra variante utilizada es la convolución 1x1 que consiste en aplicar un kernel de dimensiones unitarias, con la finalidad de comprimir los canales del *input* en uno solo.

2.4.4. Métricas utilizadas

Las métricas no solo ayudan a evaluar los resultados entregados por los modelos sino también son buenos indicadores del aprendizaje de los modelos durante su entrenamiento.

Métricas supervisadas

En estos modelos se conoce de antemano el valor de salida asociado a los *Inputs.* Suponiendo que se trata de un problema binario donde las clases son: Clase Positiva (1) y Negativa (0). Se definen los siguientes conceptos.

- Verdadero Positivo: El resultado predicho y el real son positivos.
- Verdadero Negativo: El resultado predicho y el real son negativos.
- Falso Positivo: El resultado predicho es positivo cuando el real es negativo.
- Falso Negativo: El resultado predicho es negativo cuando el real es positivo.

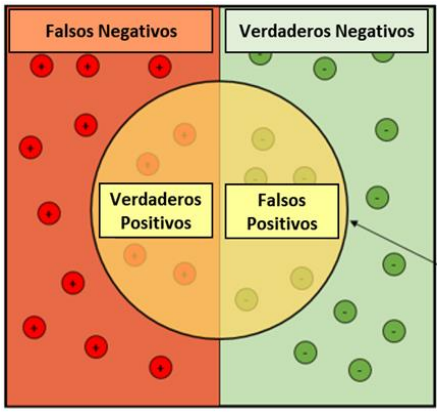


Figura 2.18: Conceptos de falsos positivos. [23]

Matriz de confusión

Una matriz de confusión se construye a partir de los valores predichos y reales. Es una forma práctica de visualizar el rendimiento de un modelo. La diagonal contiene las muestras que fueron identificadas o etiquetadas correctamente.

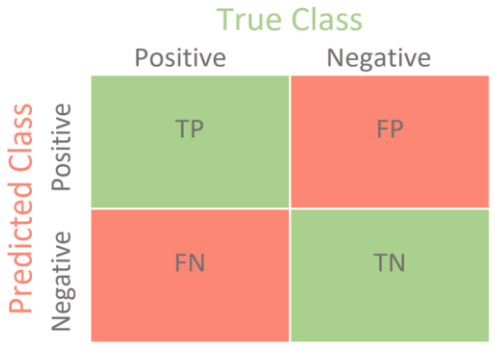


Figura 2.19: Matriz de confusión para clasificación binaria. [23]

Exactitud

La exactitud es la proporción de datos que fueron correctamente clasificados del total de datos disponibles. Es una buena métrica para problemas de clasificación bien balanceados.

$$Exactitud = \frac{VP + VN}{VP + FN + FP + VN} \quad (2.11)$$

Precisión

La precisión mide qué proporción de las muestras clasificadas como positivas corresponden efectivamente a la clase positiva. Una alta precisión indica que las muestras clasificadas positivas por el modelo tienen una alta probabilidad de ser realmente positivas.

$$Precisión = \frac{VP}{VP + FP} \quad (2.12)$$

Sensibilidad

La sensibilidad mide qué proporción de muestras positivas fueron correctamente clasificadas. Una alta sensibilidad indica que el modelo está detectando la mayoría de las muestras realmente positivas.

$$Sensibilidad = \frac{VP}{VP + FN} \quad (2.13)$$

F1-score

Es una combinación de las métricas Precisión y Sensibilidad, y por tanto es un buen indicador del desempeño general del modelo.

$$F1 - score = 2\frac{Precisión * Sensibilidad}{Precisión + Sensibilidad} \quad (2.14)$$

FPR (*False Positive Rate)*

Como su nombre lo indica es el porcentaje de falsos positivos. Se busca mantener este valor lo más cercano a cero.

$$FPR = \frac{FP}{FP + VN} \quad (2.15)$$

Métricas no supervisadas

Las métricas descritas a continuación son exclusivamente utilizadas en la evaluación de modelos de agrupamiento, y son una medida de qué tan definidos son los *clusters* formados por el modelo.

Coeficiente de silueta

El coeficiente de silueta se calcula con la siguiente fórmula.

$$s = \frac{b - a}{max(a, b)} \qquad (2.16)$$

Donde *a* es la distancia promedio de las muestras de un mismo *cluster*, y *b* la distancia promedio de cada muestra al *cluster* más cercano.

Los valores de *s* van entre 1 y -1. Un valor cercano a 0 indica que los *clusters* se superponen entre sí. Se obtienen valores altos cuando los *clusters* son densos y están bien separados [24].

Índice de Calinski-Harabaz

Corresponde a la razón entre la suma de la dispersión entre *clusters*, y la dispersión dentro de cada *cluster* para todos los *clusters*. La dispersión se define como la suma de las distancias al cuadrado. Un alto puntaje se obtiene cuando los *clusters* son densos y están bien separados [25].

Cabe mencionar que en ambas métricas se obtienen mayores puntajes para *clusters* convexos que para *clusters* basados en densidad formados por algoritmos como DBSCAN.

2.5. Mantenimiento

De acuerdo con la European standard UNE-EN13306 [26], el mantenimiento se define como la "combinación de todas las acciones técnicas, administrativas y de gestión, durante el ciclo de vida de un elemento, destinados a conservarlo o devolverlo a un estado en el cual pueda desarrollar su función requerida.

El mantenimiento juega un rol decisivo en la disponibilidad de un activo, y la forma en que se organiza el proceso de mantención viene dado por la estrategia de mantenimiento. La selección de la estrategia permite definir qué actividades de mantenimiento se deben realizar.

2.5.1. Estrategias de mantenimiento

La elección de una estrategia de mantenimiento eficiente es crucial para la reducción de costos y minimización de los tiempos de indisponibilidad de los activos, y dependerá de las necesidades y características de la empresa y su línea de producción. De acuerdo con [27] se pueden identificar 3 estrategias de mantenimiento principales, cabe destacar que el mantenimiento predictivo es sub-divido en 2, por lo que en definitiva se describen 4 estrategias de mantenimiento.

Mantenimiento Correctivo

También conocido como Mantenimiento Reactivo se fundamenta en reparar o sustituir componentes de un activo sólo cuando éste falla o ya no puede operar. La planificación consiste en identificar los componentes críticos de los equipos y asegurar un stock de recambio de estos, con la finalidad de reducir los tiempos de reacción. Si bien este tipo de mantenimiento asegura que el activo o componente usará toda su vida útil maximizando su tiempo de utilización, resulta en primer lugar potencialmente peligroso debido a que puede ocurrir una falla catastrófica y por otro lado es una estrategia poco eficiente ya que no se sabe con certeza el momento en que ocurrirá la falla además de los elevados costos y tiempos de reparación importantes.

Mantenimiento Preventivo

El Mantenimiento Preventivo o Programado se basa en realizar intervenciones de mantenimiento en intervalos de tiempo regulares incluso cuando el activo está operando en condiciones normales, con la finalidad de minimizar la probabilidad de fallas y evitar costos por detenciones no programadas. El programa de mantención debe estar basado en las propiedades de cada equipo, estimando su vida útil a partir de datos históricos de fallas y del comportamiento de los activos, de esta manera se reparan o reemplazan aquellos componentes que, según datos estadísticos, fallarían en el corto plazo. La desventaja es que el componente o activo es reemplazado antes de alcanzar su vida útil, sin embargo, este tipo de mantención es planificada con lo cual la producción no se ve interrumpida de manera imprevista.

Mantenimiento Predictivo

Con la finalidad de optimizar las estrategias de mantenimiento surge el mantenimiento predictivo y el mantenimiento basado en la condición (CBM), ambos consisten en monitorear la salud o condición de los activos, su eficiencia y otros parámetros, a través de mediciones no destructivas que van a detectar síntomas que advierten sobre una posible falla que permiten anticiparse y planificar la reparación o reemplazo del componente afectado. De acuerdo con [28], la diferencia entre mantenimiento predictivo (PM) y CBM reside solamente en la velocidad de respuesta ante la detección de una condición defectuosa. Particularmente CBM interviene en el sistema inmediatamente mientras que PM trata de predecir la vida útil remanente (RUL) del componente que presenta deficiencias, con el fin de establecer en qué momento del futuro debe realizarse la mantención. Esta diferencia se ve claramente esquematizada en la Figura 2.20 donde se resume el comportamiento de las 4 estrategias vistas anteriormente.

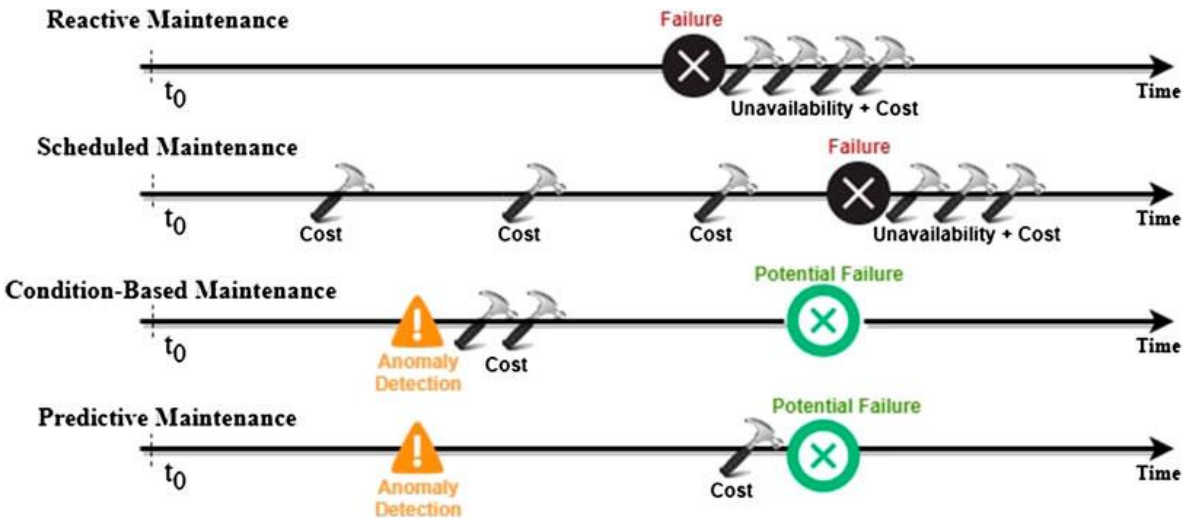


Figura 2.20: Esquema de los 4 tipos de mantenimiento [28].

En el mantenimiento predictivo basado en la condición (CBM) se utilizan distintas técnicas para medir síntomas y detectar posibles fallas tales como: Análisis de vibraciones, Ultrasonido, Termografías entre otros. Cabe destacar que estas técnicas requieren equipos de medición sofisticados, así como también de personal especializados para analizar los datos obtenidos [29]. A continuación, se describen en detalle algunas de estas técnicas.

i) Análisis de Vibraciones

Técnica que se basa en el estudio de máquinas rotativas, las cuales por naturaleza presentan vibraciones durante su operación, sin embargo, la presencia de anomalías en el nivel de vibraciones son un indicador de posibles fallas, estas anomalías se detectan analizando la señal a través de su amplitud o su descomposición en frecuencia.

Un sistema para el análisis de vibraciones consta de 3 componentes clave: un sensor, un sistema de adquisición de datos y un software para el análisis de estos datos. Además, es necesario conocer algunos datos del equipo como su velocidad de rotación, tipos de rodamientos/correas, número de álabes/palas, entre otros.

Este método permite detectar problemas tales como: desalineamiento, desbalance, resonancia, soltura mecánica, daño en rodamientos y engranajes, problemas en equipos eléctricos, problemas de montaje, problemas en bombas, problemas de bandas, etc.

ii) Termografías

Método que permite medir la temperatura de un equipo sin interrumpir su funcionamiento, lo cual se realiza utilizando una cámara termográfica que a diferencia de un termómetro permite obtener un perfil térmico de todo el equipo, pudiendo identificar puntos con temperatura anómalas. Un incremento en la temperatura puede traducirse en problemas de naturaleza electromecánica. De esta forma este método es aplicado en equipamientos mecánicos, instalaciones eléctricas y estructuras refractarias.

iii) Análisis de Ultrasonido

Este método se basa en la detección de ondas de sonido de alta frecuencia producidas por las máquinas cuando presentan un problema, como rozamiento, descargas eléctricas, pérdidas de presión o vacío. Estas ondas son de alta frecuencia, corta longitud y rápida pérdida de energía, lo que permite detectar el lugar de la falla con exactitud. Los instrumentos de detección de ultrasonido convierten la señal ultrasónica en una señal de frecuencia audible para el humano. Con esta técnica es posible detectar: fricción en máquinas rotatorias, fallas y/o fugas en válvulas, pérdidas de vacío, arco eléctrico, fugas en recintos estancos, entre otros.

iv) Análisis de Aceite

El análisis de aceite o lubricantes permite determinar el estado de operación de las máquinas. Las principales funciones del lubricante son controlar la fricción, el desgaste, la corrosión, la temperatura y eliminar las impurezas. De esta forma el análisis de aceite permite

cuantificar el grado de contaminación y degradación por medio de pruebas de laboratorio. La contaminación es un buen indicador del estado de la máquina y está relacionado con la presencia de partículas de desgaste y sustancias extrañas, las que pueden ser detectadas por medio de contaje de partículas, análisis espectrométrico, análisis infrarrojo, ferrografía analítica y ferrografía de lectura directa. Por otro lado, el grado de degradación determina el estado del lubricante cuantificando propiedades como el aspecto, color, olor, índice de acidez, viscosidad cinemática, nivel de agua, alcalinidad, detergencia y constante dieléctrica.

2.5.2. Mantenimiento 4.0

Uno de los campos de aplicación de la industria 4.0 es el mantenimiento. Las estrategias de mantenimiento predictivo, en particular las basadas en condición (*Condition Based Maintenance*) son más fáciles de implementar utilizando las herramientas tecnológicas de la industria 4.0. Con la gran cantidad de información generada por la infraestructura *IIoT* que incluye sensores, dispositivos, histórico de datos, entre otros, es posible monitorear la condición de los activos de manera continua, pero también es posible aplicar herramientas de Inteligencia Artificial, particularmente desarrollando algoritmos de *Machine Learning* que aprenden de estos datos generando nueva información asociada a la detección de anomalías, reconocimiento de fallas o modos de operación de un sistema, estimación de vida útil remanente de un activo, entre otras. Lo anteriormente descrito corresponde específicamente a la utilización de Modelos Basados en Datos (*Data-Driven Models*). Otros modelos utilizados en el contexto de mantenimiento 4.0 son los *Physical Models* y los *Knowledge-Based Models* [30].

Los Modelos Físicos se basan en el proceso físico de degradación de los componentes de un activo, modelando en base a las leyes físicas el cómo las condiciones de operación afectan la eficiencia y vida de los activos. Para ello es necesario contar con sensores que capten las variables físicas más relevantes. Este tipo de modelos tiene la ventaja de entregar una descripción física del proceso, permitiendo analizar los motivos físicos detrás de cada resultado entregado por el modelo.

Los modelos basados en conocimiento utilizan la experiencia, habilidades y conocimientos de expertos competentes en un campo determinado. Esta información es tomada por un Sistema Experto que aplica mecanismos de inferencia y razonamiento para entregar soporte y soluciones a problemas prácticos. Entre los métodos que aplican este tipo de modelos están los mecanismos basados en reglas y el *Fuzzy Logic*.

2.5.3. PHM

En el contexto de mantenimiento 4.0 y los *Data-Driven Models* surgen los sistemas PHM: *Prognostic and Health Management* cuyo objetivo es entregar un análisis exhaustivo de la condición de salud de un activo y sus componentes [31]. Estos sistemas PHM ocupan herramientas de la Ciencia de Datos, la estadística, la física y la Inteligencia Artificial con el objetivo de: detectar síntomas de una eventual falla en el sistema (Detección de Anomalías), clasificar la falla de acuerdo con su tipo (Diagnóstico), y por último predecir por cuánto tiempo puede seguir operando la máquina en presencia de esta falla (Pronóstico) [32].

El proceso PHM usa la información extraída de los datos para evaluar la salud de un componente industrial y aplicar las operaciones de mantenimiento correspondientes. En la Figura 2.21 se muestran los principales componentes que constituyen el proceso típico de PHM, destacando su rol tanto en estrategias de mantenimiento predictivo (PM) y mantenimiento basado en condición (CBM).

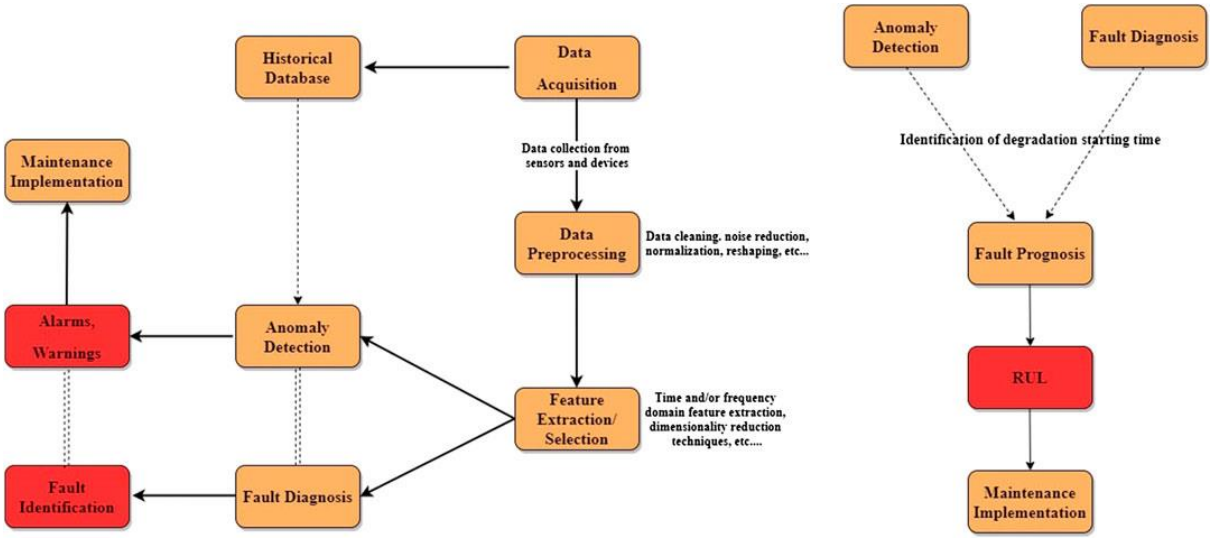


Figura 2.21: Pasos principales de un proceso típico PHM. Izquierda contexto de mantenimiento basado en la condición (CBM); Derecha contexto de aplicación en estrategias de mantenimiento predictivo [28].

A continuación, se pasa a describir más en detalle los pasos principales de un proceso típico de PHM.

1. Adquisición de Datos

Para la adquisición de datos en primer lugar es necesario definir los sistemas críticos que se quieren monitorear. Lo siguiente es identificar los componentes críticos de ese sistema en base a un análisis de fallas del sistema. Luego se debe seleccionar las variables físicas que sea pertinente monitorear en los componentes críticos ya identificados. Finalmente

seleccionar los sensores y dispositivos adecuados para la medición de los parámetros físicos, ubicarlos en la posición más apropiada y configurarlos en una frecuencia óptima de muestreo. Es importante la implementación de un sistema de comunicación que permita manejar los datos de manera *offline* y también monitorear en tiempo real.

2. Pre-Procesamiento

Los datos adquiridos por lo general no están en un formato que sea compatible con los *inputs* de los algoritmos inteligentes que se quieran aplicar, por tanto, es necesario aplicar métodos para limpiar los datos, mitigar efectos inducidos por el ruido, o reestructurar la forma de los datos.

3. Procesamiento

Los datos aún pueden contener información redundante, lo que motiva el uso de métodos de extracción de parámetros para reducir la dimensionalidad de los datos y retener solamente la información más importante. De esta manera, los datos ya están listos para ser recibidos por algoritmos inteligentes que ayuden en la toma de decisiones y que aporten información relevante sobre el estado o condición de un sistema.

4. Detección de Anomalías

Los modelos de detección de anomalías establecen de manera automática si la información que está siendo recibida presenta alguna discrepancia en comparación con el funcionamiento normal de la máquina. Los algoritmos aprenden a partir de información recopilada de activos sanos, es decir, cuando estos operan bajo condiciones normales. Cabe mencionar que una anomalía no necesariamente es una falla, a veces puede representar un nuevo modo de operación de la máquina que no estaba presente en los datos históricos.

5. Diagnóstico de Fallas

Estos modelos, además de detectar una anomalía permiten identificar la causa de esta. Están basados en datos históricos que contienen eventos de distintas condiciones defectuosas, y permiten clasificar nuevas fallas en las clases previamente definidas.

6. Pronóstico de Fallas

Los modelos de pronóstico de fallas entregan una predicción del RUL de un determinado componente. La estimación de la vida útil remanente comienza con la identificación del intervalo de tiempo en el cual se comienzan a observar los efectos de una falla, para luego inferir por cuánto tiempo puede seguir operando la máquina y establecer el intervalo de tiempo en que ocurriría la falla.

3. Metodología

Con el fin de lograr los objetivos propuestos para este trabajo de investigación, se plantea el siguiente procedimiento.

1. Primero se realiza una revisión bibliográfica de los temas pertinentes al trabajo tales como Industria 4.0, algoritmos de *Machine Learning*, Mantenimiento Predictivo, entre otros. En esta etapa también se realiza un reconocimiento del proceso productivo de la Planta Nueva Aldea poniendo especial atención a los equipos críticos posibles candidatos de estudio. Esta revisión es la base del desarrollo de antecedentes del trabajo.

2. En segunda instancia se lleva a cabo la propuesta de equipos candidatos en conjunto con miembros del área de confiabilidad de la Planta Nueva Aldea, donde además se define qué problema se quiere resolver.

3. Se realiza un análisis preliminar de los datos y la información disponible en cada equipo candidato. En este análisis se investiga qué datos hay disponibles: registro de fallas y variables de monitoreo. Los datos están almacenados en un *DataLake* IP21 y para verificar su calidad se visualizan con la herramienta *Wedge*, básicamente para saber si contienen mediciones en el periodo de interés y si las mediciones están dentro de los rangos de operación.

4. Luego se seleccionan los equipos basándose en el análisis preliminar, la relevancia del equipo en el proceso productivo, la factibilidad y utilidad de desarrollar modelos basados en aprendizaje de máquinas para resolver el problema propuesto.

5. Con los equipos seleccionados y conociendo la información disponible en cada caso, se acota el problema u objetivo específico. De esta forma se elige el tipo o familia de modelos a utilizar que son más adecuados para abordar el problema.

6. Se realiza un pre-procesamiento o tratamiento de los datos donde en primer lugar se extraen los datos a una frecuencia de muestreo y un periodo de tiempo determinados, esto definirá la cantidad de datos con que se trabaja, que es la principal limitante en los modelos utilizados (capacidad computacional limitada). Se corrigen o modifican

los registros de fallas mal ingresados, se estructuran los datos en un formato adecuado para su posterior lectura, se etiquetan los datos cuando corresponda, se limpian los datos eliminando aquellos que no son de utilidad (por ejemplo, cuando el equipo no está en servicio).

7. En el procesamiento de los datos se verifican valores faltantes, si una variable de monitoreo no contiene suficientes datos se procede a interpolar los valores o a eliminar la variable. Se elimina además información redundante o que aporta poca información mediante métodos de reducción y selección de parámetros. Si la información es insuficiente se utilizan métodos de extracción de parámetros y *feature engineering*. La separación de los datos en conjuntos de entrenamiento, validación y testeo también se realizan en el procesamiento, como también la normalización, re-etiquetado de los datos y extracción de ventanas temporales.

8. Posteriormente se entrenan diferentes modelos de la misma familia (seleccionada anteriormente) con los datos de entrenamiento. Se realizan varias iteraciones ajustando los parámetros de cada modelo con la finalidad de optimizar y mejorar el rendimiento de los modelos, es decir, que aprendan los patrones del conjunto de entrenamiento. En paralelo se evalúa su rendimiento en el conjunto de validación procurando que el modelo no se sobreajuste y sea capaz de generalizar su aprendizaje.

9. Luego de entrenar los modelos, se procede a evaluar su desempeño con datos que no han sido parte del entrenamiento, en otras palabras, datos nuevos, estos corresponden al conjunto de testeo. La evaluación se hace tanto de manera cuantitativa, mediante métricas de rendimiento, y cualitativa, usando las curvas de aprendizaje y visualizando los resultados del modelo.

10. Finalmente se analizan los resultados obtenidos, seleccionando el modelo de mejor rendimiento y que mejor resuelve el problema planteado. En base a esto se desarrollan las conclusiones del trabajo.

La metodología que se muestra en la Figura 4.1 corresponde a un esquema general para el desarrollo de los diferentes modelos. La metodología puede variar en cada caso de estudio.

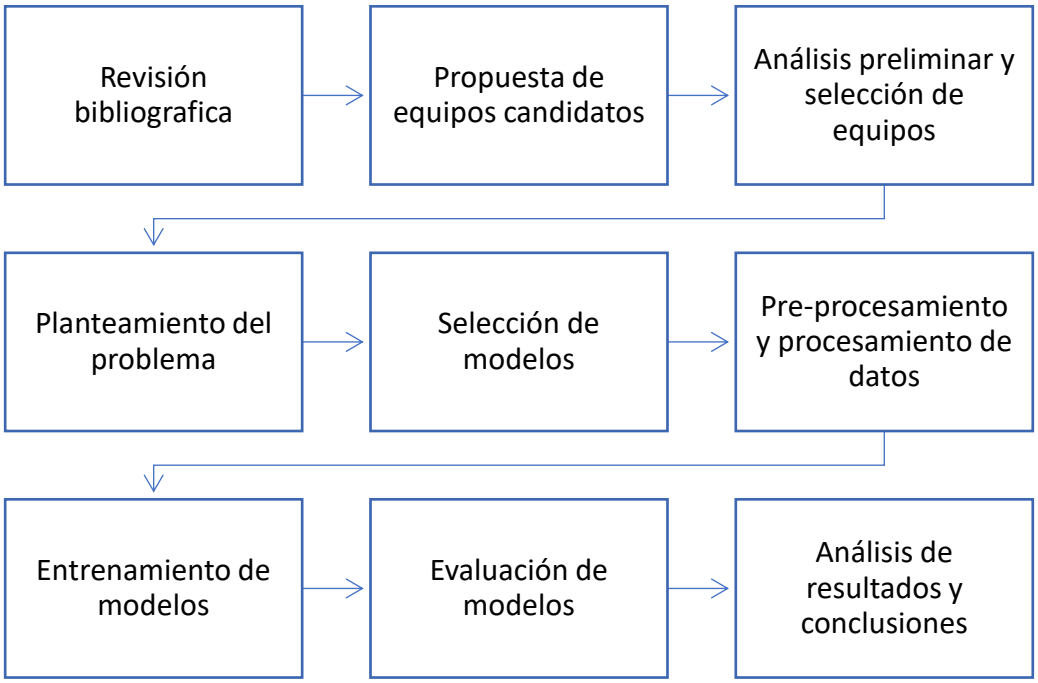


Figura 4.1. Esquema de la metodología del presente trabajo de investigación.

4. Desarrollo y discusión de resultados

4.1. Equipos candidatos

En conjunto con miembros del área de confiabilidad se propusieron equipos candidatos para el desarrollo de modelos de identificación de fallas con el objetivo de monitorear su condición. Los principales criterios de selección de los equipos de estudio fueron: la disponibilidad de los datos, la factibilidad y utilidad de desarrollar modelos de aprendizaje de máquinas para cumplir el objetivo propuesto. Si bien se consideró la relevancia y criticidad del equipo en el proceso productivo, este aspecto no fue determinante para seleccionar los casos de estudio.

A continuación, se pasa a detallar cada uno de los equipos propuestos, los objetivos, el análisis preliminar de datos e información disponible, y finalmente los equipos que fueron seleccionados como casos de estudio.

i) **Descortezador**

Descripción: Este equipo está ubicado en el área de maderas y está encargado de retirar la corteza de los troncos brutos de Pino y Eucalipto antes de ingresar al *chipper* encargado de astillar los troncos mencionados. El equipo en cuestión es un molino con forma cilíndrica cuyo eje de rotación está posicionado de manera casi horizontal. Posee una leve inclinación que permite el avance de los troncos cuando el equipo gira a bajas revoluciones. En su interior tiene aditamentos que arrastran los troncos hasta una determinada altura hasta que caen por gravedad e impactan entre ellos, lo que va desprendiendo su corteza.

Objetivo: Detectar elementos anómalos como troncos sobre diámetro, corteza no retirada, piedras y otros elementos no deseados a la salida del equipo, con el fin de evitar que estos ingresen al siguiente equipo (*chipper*). Para ello se plantea desarrollar un modelo capaz de detectar y alertar la presencia de elementos indeseados utilizando reconocimiento de imágenes.

Análisis preliminar: No se cuenta con instrumentos de adquisición de imágenes ni con datos para desarrollar el modelo, además el tiempo de desarrollo del trabajo es limitado lo que no permite instalar el instrumento y recopilar los datos suficientes para el desarrollo del modelo. Por este motivo el equipo no fue seleccionado.

ii) **Turbomáquinas**

Descripción: En la planta existe un gran volumen de turbomáquinas tales como: bombas, ventiladores, compresores, turbocompresores entre otros. Estos equipos están presentes en todos los procesos de la planta y tienen diferentes características y funciones. Las turbomáquinas propuestas son del tipo generadoras hidráulicas. Esta familia de equipos es numerosa, por lo que actualmente se prioriza en planta el monitoreo de equipos más críticos.

Objetivo: Detectar anomalías en la operación de los equipos y alertar cuando el equipo está operando en un modo de operación distinto al normal a partir de variables operacionales.

Análisis preliminar: Algunas variables de operación son monitoreadas y se cuenta con datos históricos de variables de operación registradas durante los últimos años. Dada la disponibilidad de información, se opta por considerar esta familia en el desarrollo de este trabajo.

iii) **Área de Máquina**

Descripción**:** Esta área de la planta está ubicada después del área de Blanqueo, y está encargada de la formación, secado y corte de la hoja de celulosa para la formación de los fardos.

Objetivo: Predecir el corte indeseado de hoja en los procesos de secado, e identificar las variables que más influyen en la ocurrencia del fenómeno.

Análisis preliminar: Se cuenta con datos históricos de variables de operación de los equipos del área. No se tiene un registro de los eventos de corte de hoja indeseado pero esta información puede inferirse a partir del comportamiento de ciertas variables. Esta propuesta finalmente no fue seleccionada ya que el fenómeno ocurre con una periodicidad más o menos constante, por lo que no un modelo predictivo no aportaría demasiado. Además, el corte de hoja indeseado ocurre generalmente en el equipo secador, y la causa de corte de hoja se asocia al cíclico ensuciamiento de éste.

iv) **Prensas de desplazamiento**

Descripción: Estos equipos están ubicados en el área de Fibra donde se llevan a cabo los procesos de digestión de astillas, deslignificación, lavado y blanqueo. Las prensas de desplazamiento están encargadas del lavado de la pulpa, es decir, retirar químicos reactantes disueltos en la pulpa.

Objetivo: Predecir eventos de alto torque e identificar variables que más influyen en la ocurrencia del fenómeno.

Análisis preliminar: Se cuenta con un registro histórico de eventos de fallas en las prensas de la línea 1 de fibra. También se cuenta con datos históricos de algunas variables de operación en los periodos de interés.

v) **Caldera recuperadora**

Descripción: Este sistema del área de Licor está encargado de generar vapor de alta presión para el accionamiento de las unidades turbogeneradoras responsables de energizar todos los procesos de la planta. La energía térmica para generar vapor de agua se consigue mediante la combustión de Licor Negro concentrado, proceso a partir del cual se obtiene además Licor Verde como sub-producto.

Objetivo: Predecir eventos de falla por ensuciamiento de tubos de la caldera e identificar variables que más influyen en la ocurrencia del fenómeno.

Análisis preliminar: Se tiene conocimiento de la cantidad de eventos históricos por ensuciamiento de tubos, pero no se cuenta con el registro sistemático de estos eventos. Se tiene acceso a datos de monitoreo de algunas variables de operación del sistema. Esta propuesta no fue seleccionada debido a la falta de información sobre los eventos de falla.

En base a lo anteriormente expuesto fueron seleccionadas dos propuestas: Turbomáquinas y Prensas de desplazamiento. Para facilitar la lectura y comprensión del informe, el desarrollo y resultados son presentados en dos secciones diferentes correspondientes a cada caso de estudio.

4.2. Turbomáquinas

Como se mencionó en la sección anterior, el objetivo específico planteado es desarrollar modelos para detectar anomalías en la operación de los equipos y alertar cuando el equipo está operando en un modo de operación distinto al normal a partir de variables operacionales

4.2.1. Adquisición de datos

En cuanto a los datos disponibles, no se tiene un registro histórico de fallas, pero sí se cuenta con datos de monitoreo. Las variables de monitoreo disponibles corresponden a la velocidad del equipo y potencia del motor. En algunos casos particulares en lugar de la velocidad se tiene la variable flujo.

Se estudiaron un total de 24 turbomáquinas, entre ellos bombas y ventiladores. En la tabla 4.1 se muestra un resumen de los equipos.

Tabla 4.1: Resumen de turbomáquinas estudiadas

Turbomáquinas	Cantidad de Equipos
Bombas	10
Ventiladores (Extractores, inductores, sopladores, etc.)	14

Los datos de monitoreo se obtienen desde IP21 utilizando la plataforma *Wedge* y son exportados en archivos de formato Excel. Antes de la extracción, se realizó el análisis preliminar de los datos, donde básicamente se visualizaron los datos en la plataforma de *Wedge* para determinar si los datos son adecuados para desarrollar un modelo, y en base a esto posteriormente plantear el problema.

Como ejemplo, se muestran las variables velocidad y potencia de un período de aproximadamente 2 meses de operación del ventilador de tiro inducido del Horno de Cal (Figura 4.1). La escala de colores indica la temporalidad de los datos, donde el color rojo corresponde a los registros más recientes.

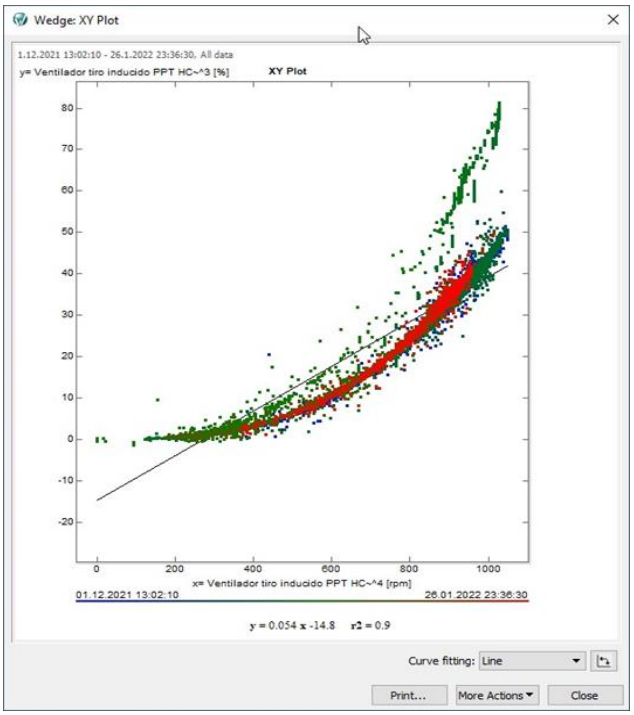


Figura 4.1: Datos de monitoreo del VTI del Horno de Cal. En el eje X la velocidad y en el eje Y la potencia.

Se observa que el equipo opera la mayor parte del tiempo en una región bien definida, sin embargo, en cierto periodo estuvo operando fuera de esta. La hipótesis es que las turbomáquinas operan regularmente en una región "normal" y que estos mismos operan fuera de dicha región cuando existe alguna anormalidad, ya sea en las condiciones de operación o asociadas a la integridad del equipo.

4.2.2. Planteamiento del problema y selección de modelos

Para identificar la región de operación normal de cada turbomáquina se plantea utilizar los siguientes modelos:

i) **Agrupamiento**

Como no se tiene conocimiento a priori de cuales datos corresponden a un funcionamiento normal y cuales no, se propone utilizar algoritmos no supervisados que permitan agrupar los datos y de esta forma etiquetarlos. En principio sólo se trabajaría con dos etiquetas: "Normal" y "Anormal" ya que no existe información *en línea* sobre modos de falla o modos de operación de los equipos.

ii) **Detección de novedades**

Los algoritmos de agrupamiento no tienen la capacidad de clasificar datos nuevos (al menos los implementados en una primera instancia), y por tanto no serían útiles para el monitoreo de la condición de los equipos. De esta forma, se plantea entrenar algoritmos de detección de novedades con los datos de la clase "Normal", para posteriormente determinar si un nuevo dato es "Normal" o "Anormal".

En una primera versión se utilizaron estos dos tipos de modelos en conjunto. Después se encontró otro modelo de agrupamiento (HDBSCAN) capaz de clasificar nuevos datos a partir de las clases identificadas previamente en los datos de entrenamiento.

4.2.3. Procesamiento de datos

A continuación, se describen las distintas acciones de procesamiento realizadas con los datos con el fin de preparar los datos para el entrenamiento de los modelos.

Las distintas versiones de los modelos desarrollados requieren distintos procedimientos para el tratamiento de datos, motivo por el cual las acciones realizadas se describen de forma individual y sin un orden particular.

Etiquetado de los datos

Como se mencionó en la sección anterior, los datos se etiquetan con la ayuda de algoritmos de agrupamiento como "normal" y "anormal". Se evaluaron diferentes algoritmos para realizar esta tarea y los resultados son mostrados en la sección 4.2.4.

Condición de servicio

Para mejorar el desempeño de los modelos y algoritmos se eliminan aquellos datos que corresponden a detenciones o partidas del equipo ya que no son parte de la operación normal. De esta manera se introduce la variable "En Servicio", que toma el valor 1 cuando la carga del motor eléctrico supera el 10%, y el valor 0 en caso contrario.

Datos faltantes

El manejo de datos faltantes se hizo de la siguiente manera: Si la cantidad de datos faltantes supera el 10% del total de datos, se busca otro período en que no exista tal cantidad de datos faltantes. Si no se supera el umbral, los datos son rellenados mediante interpolación lineal.

Sólo en algunos casos aislados se encontraron datos faltantes, y en una proporción muy baja, de manera que fue necesaria la interpolación lineal de unos pocos datos.

Normalización

Antes de entrenar los modelos se normalizaron los datos a partir de los rangos de operación de cada equipo, utilizando la normalización max-min mostrada en la ecuación 4.1.

$$X_{norm} = \frac{X - min(x)}{max(x) - min(x)} \quad (4.2)$$

4.2.4. Resultados

Como se mencionó anteriormente se realizaron 2 versiones para el desarrollo de modelos. La versión I corresponde a un modelo de agrupamiento para el etiquetado de los datos en conjunto con un modelo de detección de novedades para el monitoreo del equipo.

Selección Modelo de Agrupamiento

Para la selección del modelo adecuado se utilizaron los datos de una turbomáquina en particular, el ventilador de tiro inducido del Horno de Cal. Estos datos corresponden a un período de aproximadamente 2 meses (01/12/21 al 26/01/22) con una frecuencia de muestreo equivalente a 1 muestra cada 10 minutos o 1/600 [Hz]. Se elige esta frecuencia de muestreo luego de comprobar que con frecuencias más altas (1/60 [Hz] y 1/10 [Hz]) algunos modelos superaban la capacidad computacional durante el entrenamiento. En cuanto al procesamiento de los datos solo se realizó la Normalización max-min.

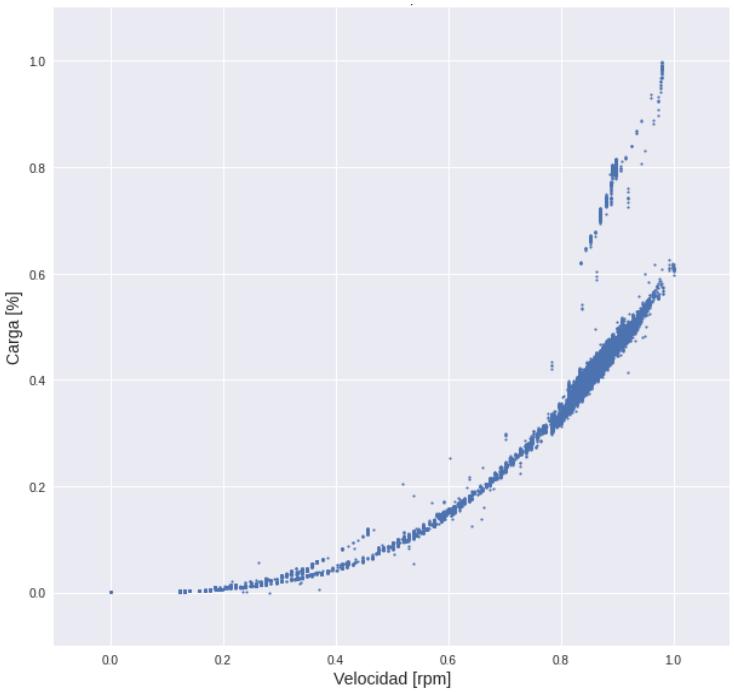


Figura 4.2: Datos Normalizados Ventilador Tiro Inducido Horno de Cal.

Se evaluaron diferentes algoritmos de agrupamiento, los cuales fueron descritos en el capítulo de Antecedentes. Lo que se busca con estos algoritmos es identificar los dos conjuntos de datos que se aprecian en la figura 4.2, donde el conjunto inferior correspondería a la operación normal del ventilador, mientras que el conjunto superior sería un comportamiento anómalo.

Finalmente, para la evaluación de cada algoritmo y selección de aquellos con mejor desempeño se calcularon métricas y se visualizaron los datos etiquetados, considerando el criterio experto.

K-Means

El único parámetro ajustable de este algoritmo es *n_clusters*, y para estimar su valor se hace uso del método del codo.

Figura 4.3; Número de clusters en función de la Inercia.

Como se observa en la figura 4.3, la formación del codo ocurre cuando el número de *clusters* es igual a 2. En otros algoritmos de agrupamiento se obtuvieron resultados más satisfactorios con 4 *clusters*, de modo que también se evaluó con ese valor.

Tabla 4.2: Métricas obtenidas con algoritmo K-Means.

Métrica	*N° clusters*	
	2	4
Coeficiente de Silueta	0.80	0.60
Índice Calinski-Harabaz	215262.75	256136.00

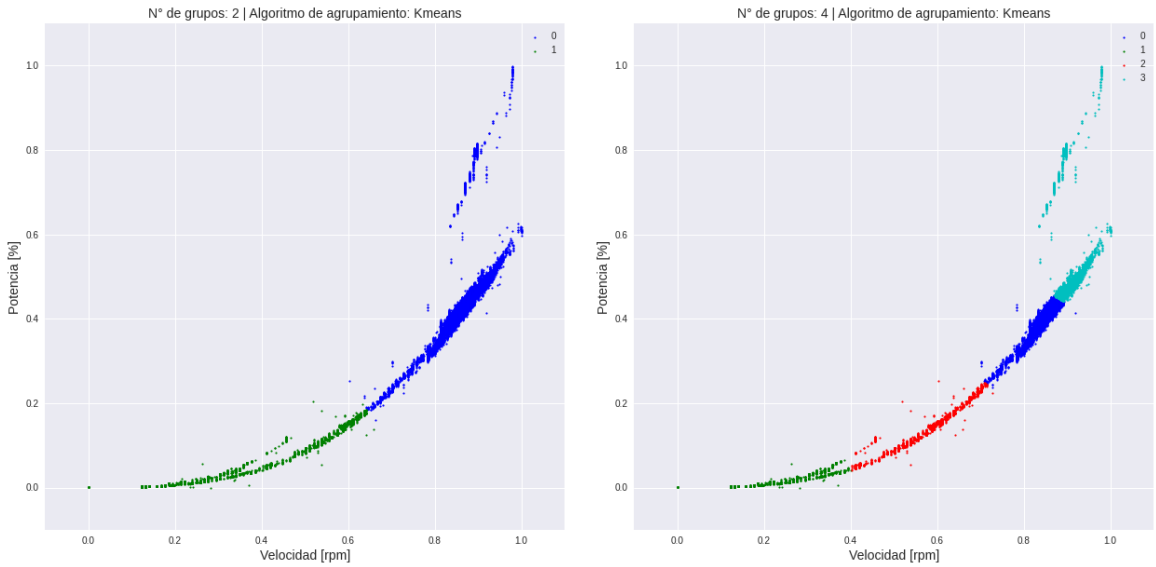


Figura 4.4: Agrupamiento realizado por algoritmo K-Means. Iizquierda n_clusters: 2; Derecha n_clusters: 4.

En la Figura 4.4 se observa que en ninguno de los dos casos el algoritmo logra agrupar de forma independiente los conjuntos superior e inferior que se quieren identificar.

Mean Shift

Se evaluó con distintos valores del parámetro *Bandwidth* hasta obtener un resultado satisfactorio. A continuación, se muestran los resultados obtenidos con algunas iteraciones del parámetro *Bandwidth*.

Tabla 4.3: Métricas obtenidas con algoritmo Mean Shift.

Métrica	*Bandwidth*		
	0.1	0.2	0.3
Coeficiente de Silueta	0.66	0.75	0.8
Índice Calinski-Harabaz	150416.35	230037.27	215016.60

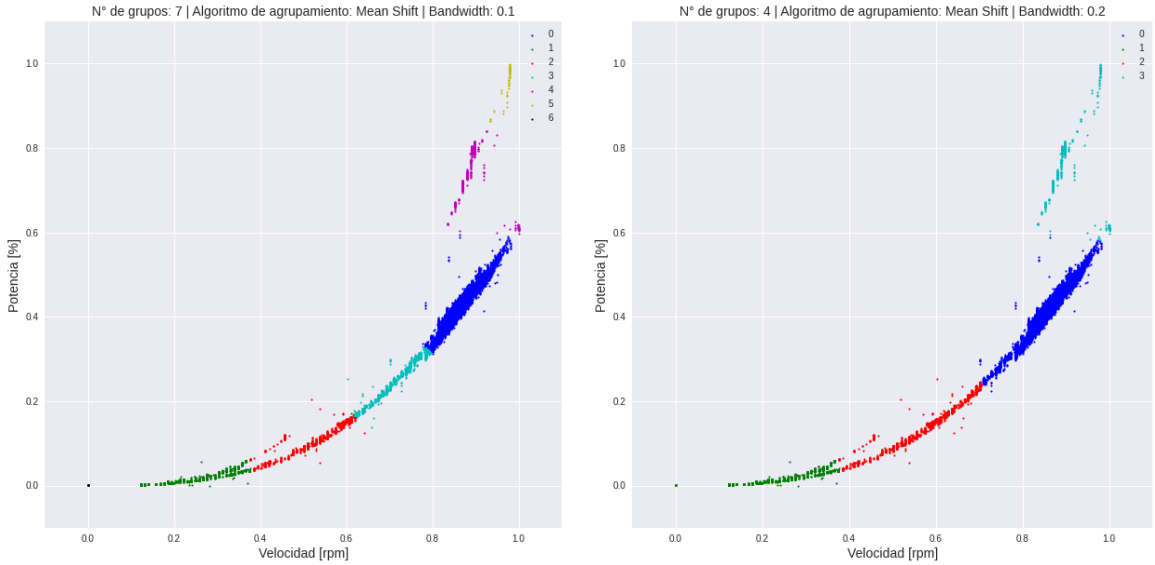


Figura 4.5: Agrupamiento realizado por algoritmo Mean Shift. Izquierda Bandwidht: 0.1; Derecha bandwidth: 0.2.

Como se aprecia en la Figura 4.5, con un *Bandwidth* de 0.2 se obtuvo un agrupamiento más acorde al resultado esperado, ya que el algoritmo agrupa gran parte del conjunto superior en un solo *cluster*, aunque en el caso del conjunto inferior es dividido en 3 *clusters* diferentes.

GMM

En base a lo obtenido con el método del codo, el parámetro *n_components* toma el valor 2. También se evalúa con 4 componentes y se comparan los resultados.

Tabla 4.4: Métricas obtenidas con algoritmo Gaussian Mixture Models.

Métrica	*N° components*	
	2	4
Coeficiente de Silueta	0.80	0.75
Índice Calinski-Harabaz	212797.76	198035.25

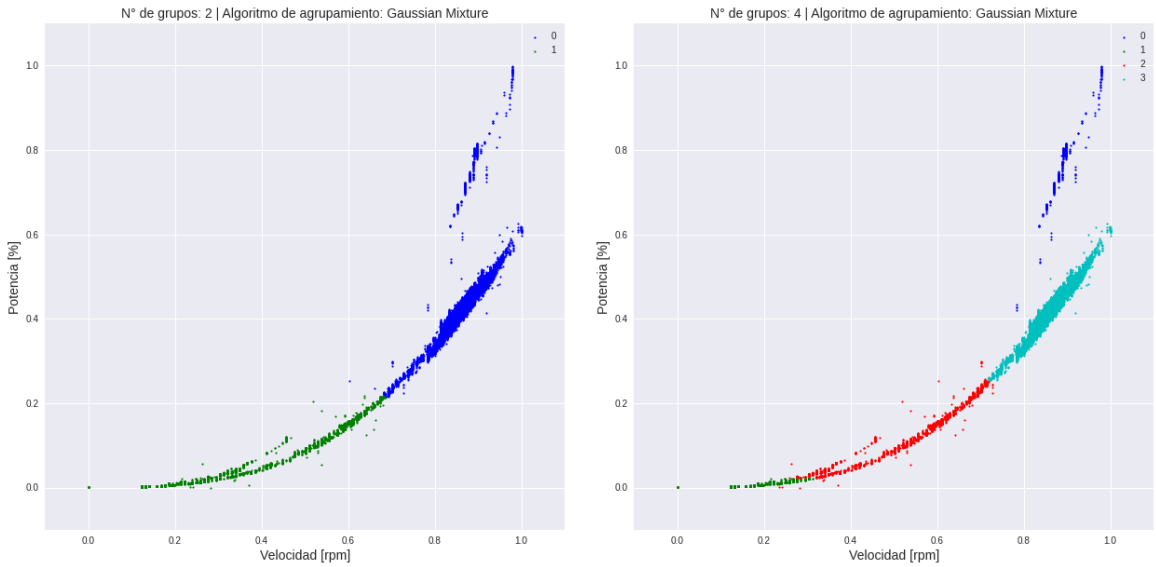


Figura 4.6: Agrupamiento realizado por algoritmo GMM. Izquierda n_components: 2; Derecha n_components: 4.

Para 4 componentes (derecha de la Figura 4.6), se observa que el conjunto superior es agrupado de forma correcta en el *cluster* de color azul, sin embargo, al igual que con el algoritmo Mean Shift el conjunto inferior es separado en 3 *clusters* diferentes.

DBSCAN

El parámetro *min_samples* se estima como 2 veces la cantidad de variables de entrada, de modo que el valor se fija en el valor 4. En cuanto al parámetro *epsilon*, se iteró con distintos valores. La tabla 4.5 y la Figura 4.7 muestran los resultados de algunas iteraciones.

Tabla 4.5: Métricas obtenidas con algoritmo DBSCAN.

Métrica	*Epsilon*		
	0.03	0.05	0.3
Coeficiente de Silueta	-0.55	0.06	0.49
Índice Calinski-Harabaz	365.93	610.39	77.33

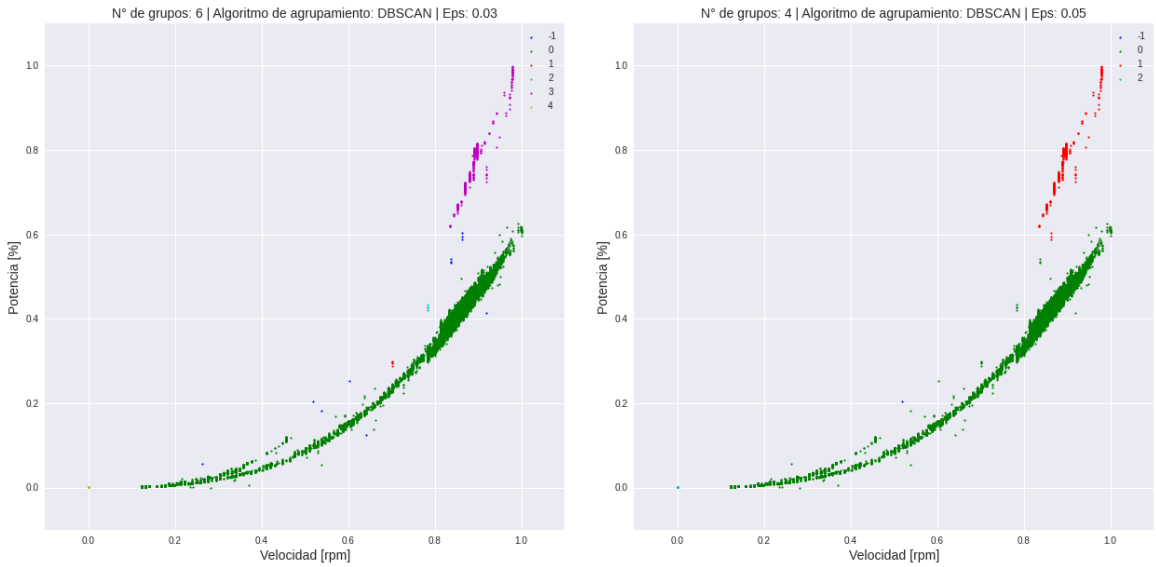


Figura 4.7: Agrupamiento realizado por algoritmo DBSCAN. Izquierda epsilon: 0.03; Derecha epsilon: 0.05.

En la imagen de la derecha de la Figura 4.7 se observa que para un *epsilon* de 0.05 se obtiene exactamente el resultado esperado: el *cluster* de color rojo contiene todos los datos del conjunto superior, mientras que el *cluster* verde contiene al conjunto inferior. Para un *epsilon* de 0.03 también se obtiene el resultado buscado, aunque aparecen más *clusters* debido a la presencia de algunos grupos de puntos aislados.

HAC

Se usa como referencia lo indicado por el método del codo, por lo que el parámetro *n_clusters* toma el valor 2, aunque también se evaluó con 4 *clusters.*

Tabla 4.6: Métricas obtenidas con algoritmo Hierarchical Agglomerative Clustering.

Métrica	*N° clusters*	
	2	4
Coeficiente de Silueta	0.77	0.75
Índice Calinski-Harabaz	170224.32	218756.44

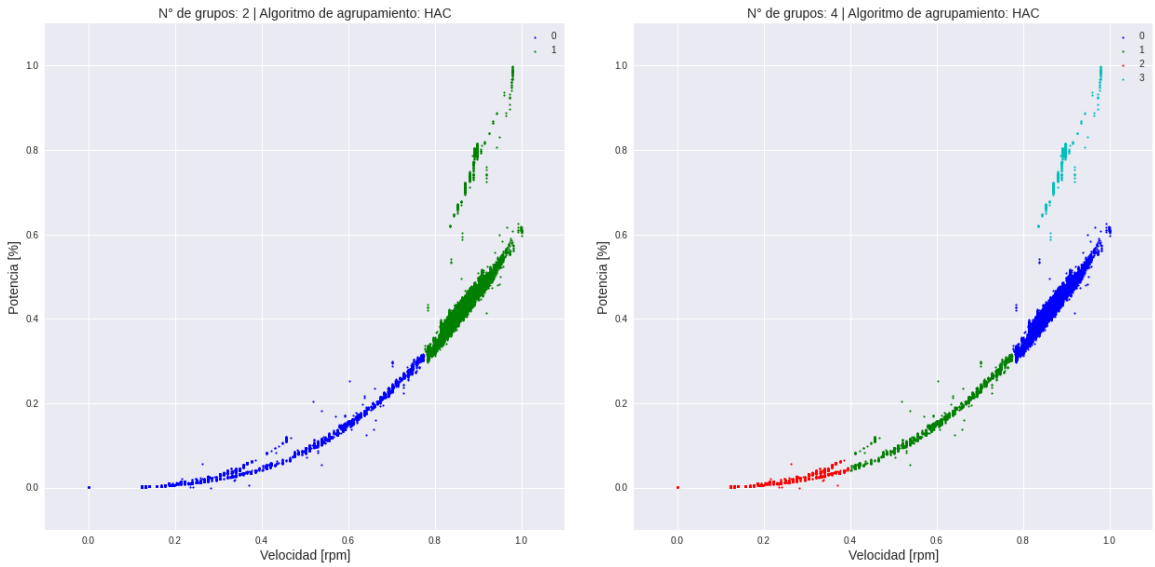


Figura 4.8: Agrupamiento realizado por algoritmo HAC. Izquierda n_clusters: 2; Derecha n_clusters: 4.

Similar a lo ocurrido con el algoritmo GMM y Mean Shift, vemos en la imagen de la derecha de la Figura 4.8 que el conjunto superior es agrupado correctamente, pero con el inconveniente que el conjunto inferior es agrupado en 3 *clusters* distintos.

De acuerdo con los resultados anteriores, el algoritmo DBSCAN es seleccionado para el etiquetado de los datos ya que es el único que agrupa correctamente los conjuntos de datos que se querían identificar en un principio. El *cluster* inferior correspondiente a la operación "Normal" y el *cluster* superior de operación "Anormal".

Si bien es cierto para un valor de *épsilon* igual a 0.05 (Figura 4.7) el algoritmo identifica un total de 4 *clusters*, dos de ellos son descartados: El *cluster* de color azul corresponde al ruido (puntos aislados), mientras que el *cluster* de color celeste corresponde a datos en que el equipo está detenido.

Por otro lado, el algoritmo DBSCAN fue el peor en cuanto a las métricas Coeficiente de Silueta e Índice Calinski-Harabaz. Este resultado tiene explicación ya que estas métricas suelen entregar peores puntajes cuando se trata de agrupamientos basados en densidad. Por este motivo, se concluye que las métricas tradicionales para agrupamiento no son las más adecuadas para la selección del algoritmo más apropiado en el etiquetado de los datos.

Etiquetado de los datos

Una vez seleccionado el algoritmo DBSCAN para el etiquetado de los datos, se procede a etiquetar los datos de todas las turbomáquinas. Los datos corresponden a un

periodo de 4 meses de monitoreo de las variables velocidad (en algunos casos flujo) y potencia o carga.

Los datos son procesados previamente. Se realizan todas las etapas de procesamiento descritas en la sección anterior, esto es: normalización max-min, condición de servicio y datos faltantes.

Para mostrar el procedimiento de etiquetado de datos se presentan los resultados de la bomba de licor verde 303. El procedimiento aquí presentado es el mismo que se utilizó para todas las turbomáquinas, cuyos resultados se presentan en el Anexo.

Para la selección del *epsilon* "óptimo", se evaluaron 8 valores diferentes y se calcularon las métricas: Coeficiente de Silueta y Indice Calisnki-Harabaz para cada *epsilon*. Luego son seleccionados los valores de *epsilon* que maximizan cada una de las métricas y se grafica el agrupamiento realizado por el algoritmo para cada *epsilon*, seleccionando aquel que entregue un resultado más coherente.

Tabla 4.7: Métricas obtenidas para distintos valores de épsilon.

Métrica	*epsilon*							
	0.4	0.2	0.1	0.08	0.06	0.04	0.02	0.01
Coeficiente de Silueta	0.73	0.73	0.73	0.67	0.66	0.34	0.34	0.08
Calinski-Harabaz	15.28	15.28	15.28	23.96	29.12	474.28	515.49	273.49

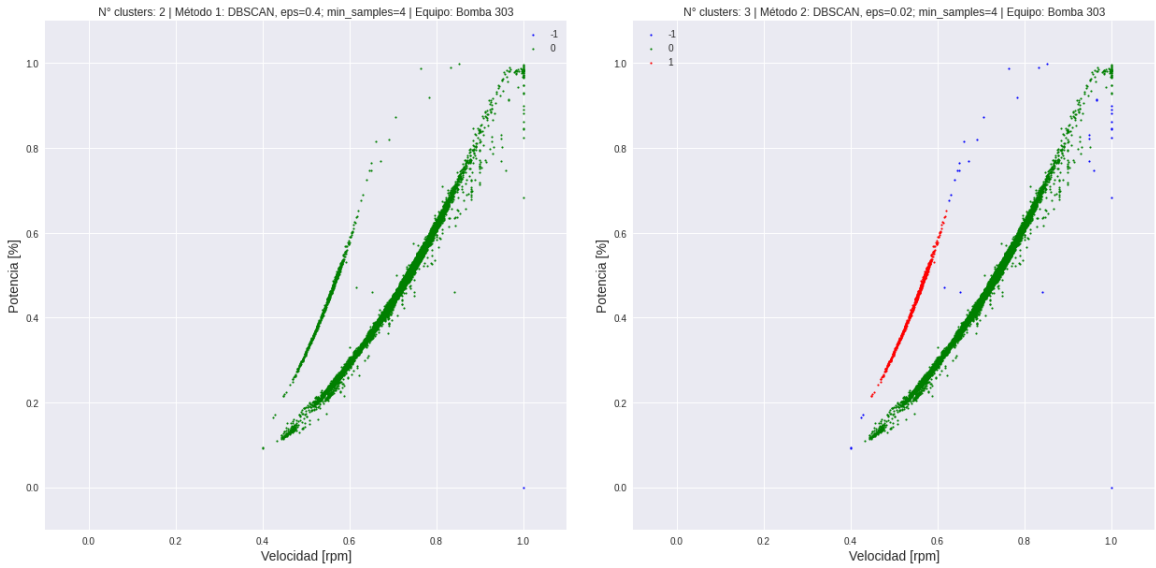


Figura 4.9: Agrupamiento realizado por algoritmo DBSCAN; Izquierda épsilon: 0.4; Derecha épsilon: 0.02.

La imagen de la derecha de la Figura 4.9 muestra claramente un mejor agrupamiento de los datos, donde se identifican 2 *clusters*, y algunos puntos aislados (ruido).

Posteriormente se re-etiquetan los datos de tal manera que solo existan dos clases, normal y anormal. El criterio utilizado es el siguiente: dado que el equipo está operando en forma normal la mayor parte del tiempo, la clase mayoritaria será la clase normal, y el resto de las clases (en caso de haber más de una) se les asigna la clase anormal. El resultado del re-etiquetado se muestra en la Figura 4.10.

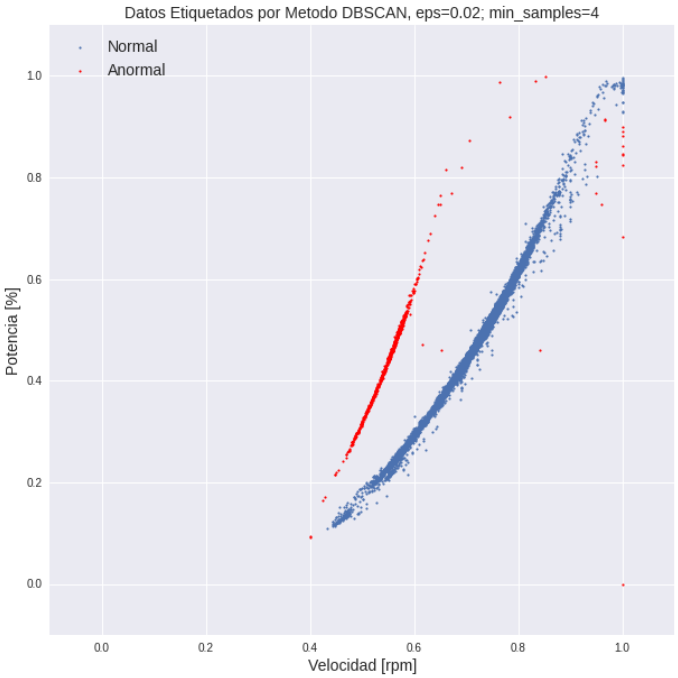


Figura 4.10: Re-etiquetado de los datos en clases normal y anormal.

Algoritmos de detección de novedades

Los algoritmos de detección de novedades utilizados son los descritos en el capítulo de antecedentes. El entrenamiento de estos algoritmos se hace con los datos de la clase "Normal". Todos los algoritmos se entrenaron con un valor 0.01 para el parámetro *contamination*, a excepción del algoritmo One Class Support Vector Machine que no usa este parámetro ajustable.

Posteriormente se valida el modelo con los datos de las clases "Normal" y "Anormal", y se compara el etiquetado realizado por el algoritmo DBSCAN con la clasificación realizada por cada modelo de detección de novedades.

La evaluación de los modelos se hace mediante el cálculo de las métricas: Exactitud, Precisión, Sensibilidad, F1-score y visualización del agrupamiento y la Frontera de decisión.

Elliptic envelope

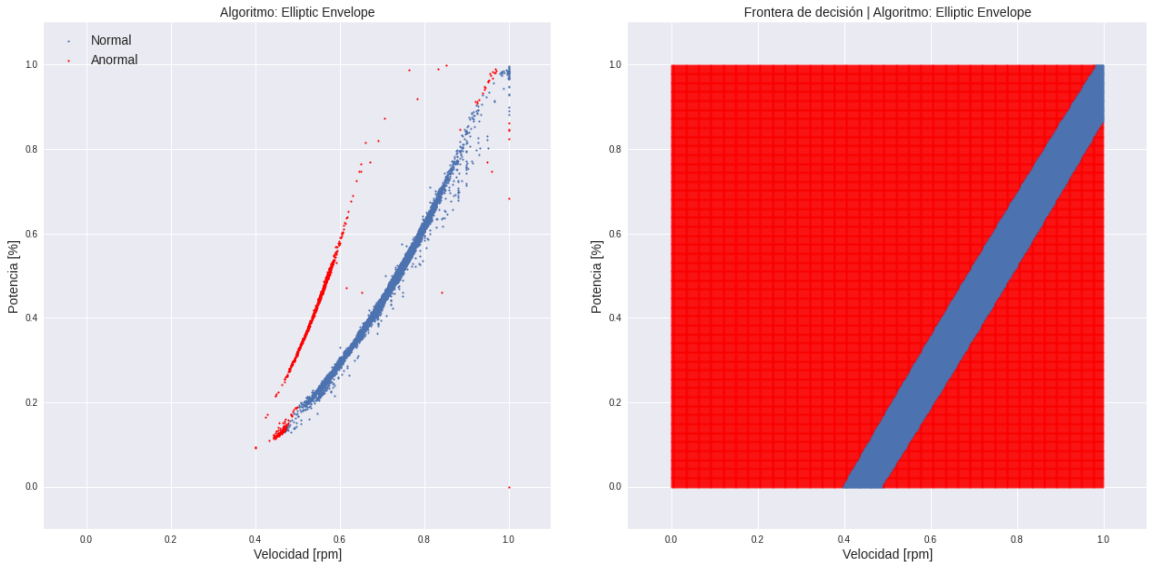


Figura 4.11: Resultado algoritmo Elliptic Envelope; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

Como muestra la izquierda de la Figura 4.11, los resultados de clasificación del algoritmo Elliptic Envelope son similares al agrupamiento obtenido con DBSCAN, sin embargo, un buen número de datos normales (negativos) están mal clasificados como datos anormales (positivos). La precisión de un 80% (Tabla 4.8) también es evidencia de la presencia de falsos positivos. Estos resultados se explican por la forma en que opera el algoritmo, el cual en la práctica ajusta una elipse al conjunto de datos normales, elipse que se observa claramente en la frontera de decisión (Figura 4.11).

Tabla 4.8: Métricas obtenidas con algoritmo Elliptic Envelope.

Exactitud	Precisión	Sensibilidad	F1-score
0.9898	0.8024	0.9883	0.8857

Isolation Forest

Se iteró con distintos valores del parámetro ajustable *n_estimators* como muestra la Tabla 4.9, para luego seleccionar aquel valor que maximiza la métrica F1-score.

Tabla 4.9: F1-score obtenido para distintos valores de n_estimtors.

n_estimators	10	50	100	500	1000
F1-score	0.0447	0.0396	0.0497	0.0398	0.0419

Como se observa en la tabla 4.9, un *n_estimators* igual a 100 maximiza el F1-score, sin embargo, es importante notar que los valores obtenidos para esta métrica son demasiado bajos, indicando que el desempeño del algoritmo es bastante pobre.

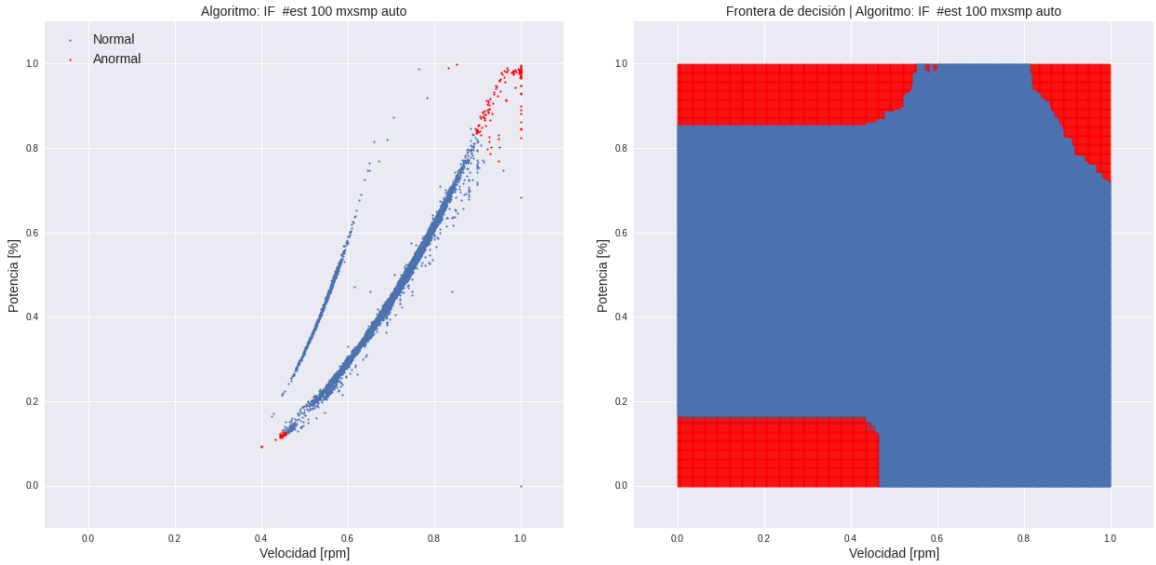


Figura 4.12: Resultado algoritmo Isolation Forest; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

La izquierda de la Figura 4.12 muestra la clasificación realizada por el algoritmo, donde claramente notamos que no se corresponde con el resultado de agrupamiento realizado por el algoritmo DBSCAN. Pareciera que el algoritmo sólo clasifica anormales a las muestras más periféricas, lo cual tiene sentido con la forma operación del algoritmo, puesto que estas muestras son más fáciles de aislar. Debido a que las muestras anormales están muy bien agrupadas, son más difíciles de aislar y por tanto el algoritmo les asigna la clase normal.

Tabla 4.10: Métricas obtenidas con algoritmo Isolation Forest.

Exactitud	Precisión	Sensibilidad	F1-score
0.9517	0.9517	0.0267	0.0419

La baja sensibilidad mostrada en la Tabla 4.10 reafirma el hecho de que el algoritmo no está detectando las muestras positivas.

Local Outlier Factor

Se evaluaron diferentes valores del parámetro ajustable *n_neighbors*, como muestra la Tabla 4.11, donde se encontró que con 5 vecinos se obtiene el máximo F1-score.

Tabla 4.11: F1-score obtenido para distintos valores de n_neighbors.

n_neighbors	5	15	30	50	70	120
F1-score	0.9273	0.9010	0.8959	0.8875	0.8832	0.8819

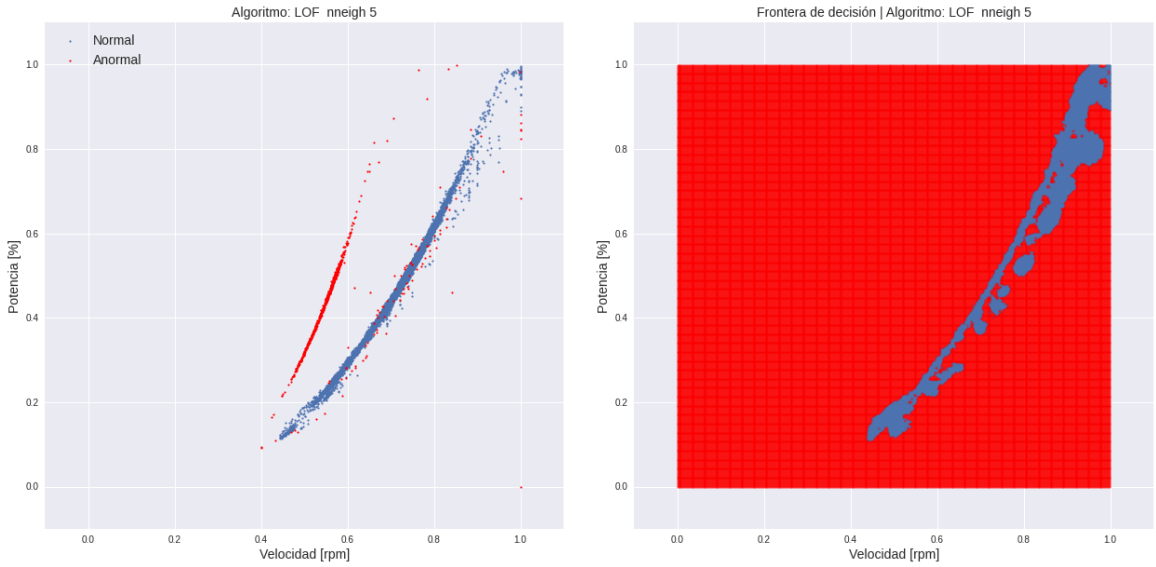


Figura 4.13: Resultado algoritmo Local Outlier Factor; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

Los resultados de este algoritmo son bastante certeros como se observa a la izquierda de la Figura 4.13. La clase positiva es correctamente clasificada en su mayoría, como también lo indica la alta sensibilidad alcanzada (~99%), y tampoco existen muchos falsos positivos (precisión del 87%). Por otro lado, al observar la frontera de decisión (derecha Figura 4.13) se aprecia que la región normal se ajusta correctamente, aunque presenta discontinuidades, cuando lo que se busca es una región normal que sea continua para evitar falsas alarmas una vez que el modelo sea implementado.

Tabla 4.12: Metricas obtenidas con algoritmo Local Outlier Factor.

Exactitud	Precisión	Sensibilidad	F1-score
0.9938	0.8733	0.9883	0.9273

OC-SVM

Se evaluaron distintos valores del parámetro *nu*. En cuanto a los otros parámetros ajustables, el parámetro *gamma* fue ajustado en el su valor predeterminado *auto*, mientras que la función kernel utilizada es *Radial Basis Function*.

Tabla 4.13: F1-score obtenido para distintos valores de nu.

nu	0.5	0.2	0.1	0.08	0.06	0.04	0.01
F1-score	0.1417	0.1268	0.0877	0.0838	0.0747	0.0623	0.0317

Un *nu* de 0.5 maximiza la métrica F1-score, sin embargo, se debe destacar que los valores obtenidos de la métrica son bastante bajos, evidenciando un bajo desempeño del algoritmo.

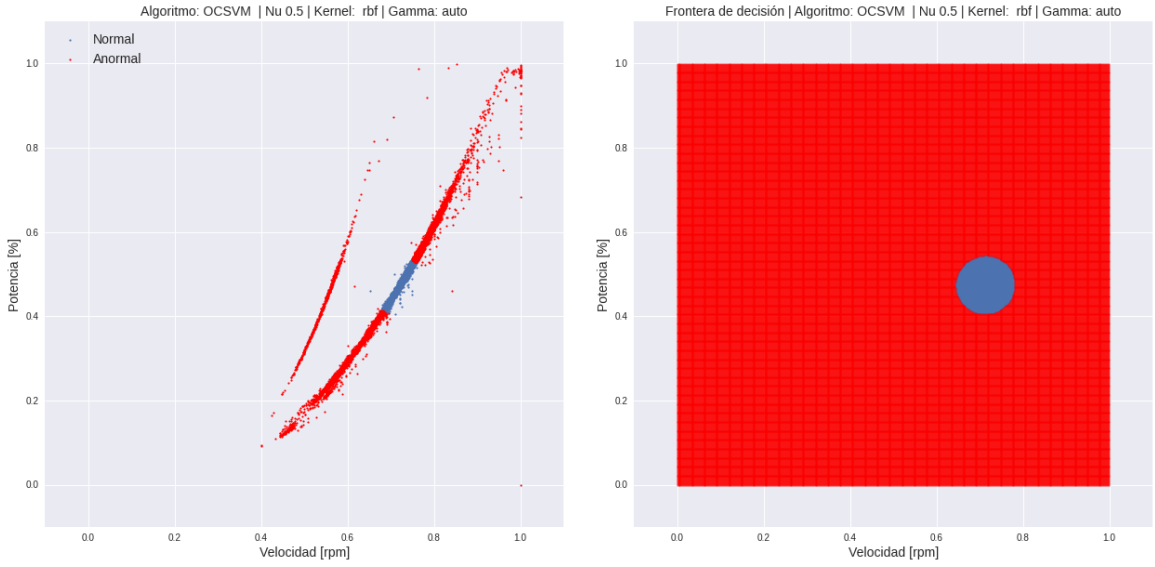


Figura 4.14: Resultado algoritmo OC-SVM con kernel rbf; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

La Figura 4.14 muestra que el algoritmo clasifica anormales la mayoría de las muestras, de modo que si bien logra identificar todas las muestras positivas (sensibilidad cercana al 100%), lo hace a costa de tener muchos falsos positivos, es decir, la mayor parte de las muestras clasificadas anormales en realidad son normales (sensibilidad de 0.1%). Este resultado indicaría que la transformación no lineal realizada por la función kernel *rbf* no es adecuada para los datos que se tienen.

Tabla 4.14: Métricas obtenidas con algoritmo OC-SVM con kernel rbf.

Exactitud	Precisión	Sensibilidad	F1-score
0.5198	0.0763	0.9983	0.1417

De igual manera se iteró con los valores anteriormente expuestos del parámetro *nu*, pero con un kernel polinomial. El F1-score se maximizan para un *un* de 0.2.

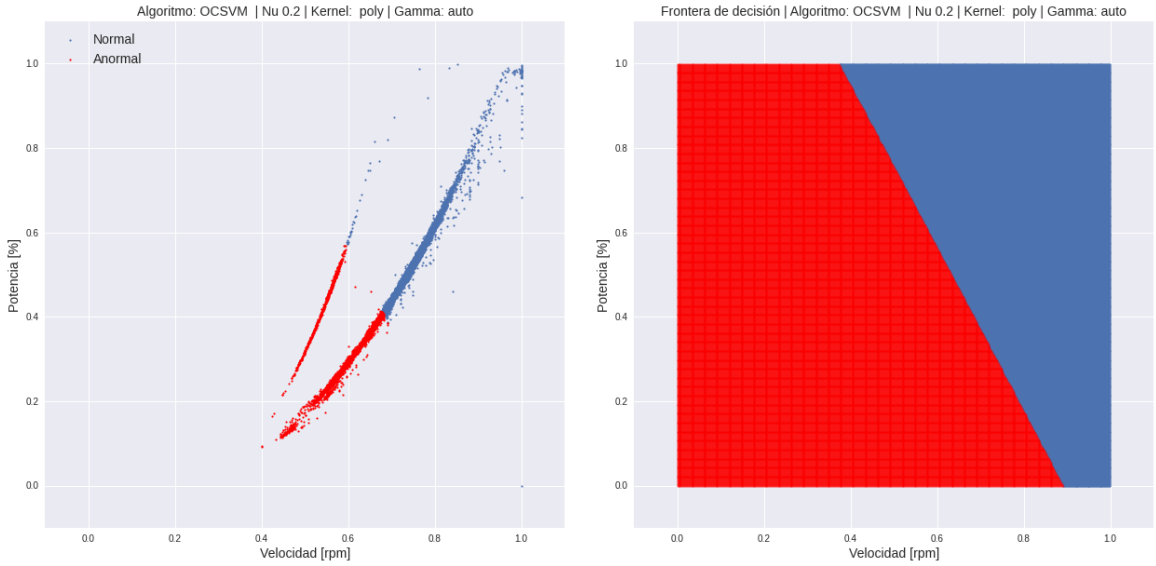


Figura 4.15: Resultado algoritmo OC-SVM con kernel polinomial; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

La derecha de la Figura 4.15 muestra que la frontera de decisión es lineal. El sentido común ubicaría esta frontera con una orientación tal, que ambos grupos de datos serían perfectamente separables, sin embargo, el algoritmo no es capaz de establecer dicha

separación. Esto es más atribuible a un error en el entrenamiento o ajuste de parámetros más que a una falencia del algoritmo.

Tabla 4.15: Métricas obtenidas con algoritmo OC-SVM con kernel polinomial.

Exactitud	Precisión	Sensibilidad	F1-score
0.8045	0.1589	0.9133	0.2707

De forma análoga se evaluó con un kernel sigmoidal, donde el F1-score se maximiza para un *nu* de 0.2.

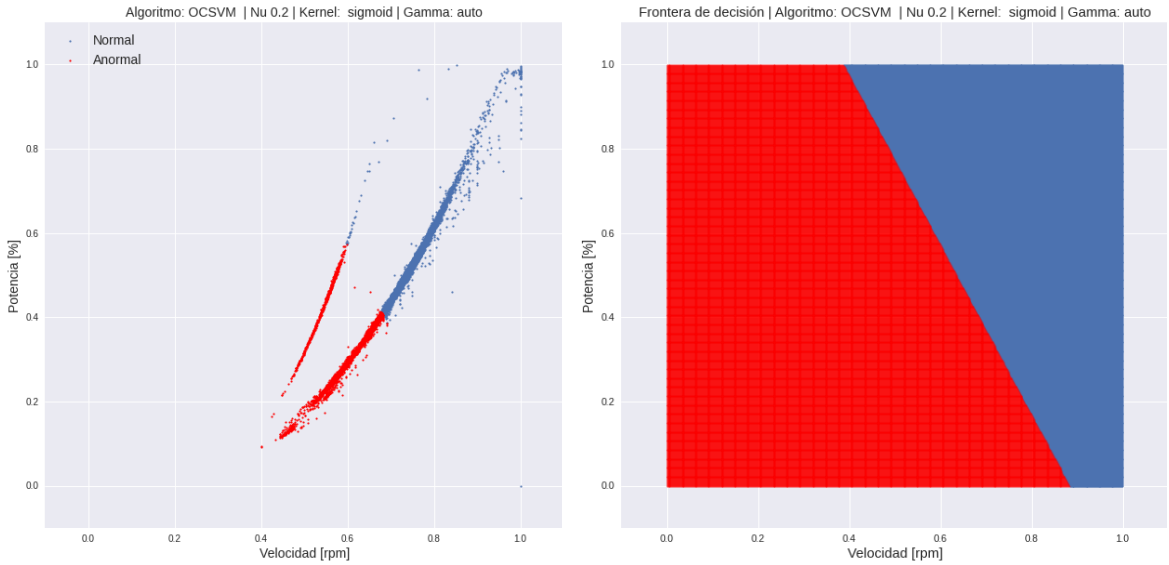


Figura 4.16: Resultado algoritmo OC-SVM con kernel sigmoidal; Izquierda: Clasificación de datos; Derecha: Frontera de decisión.

La Figura 4.16 muestra resultados muy idénticos a los obtenidos con la función kernel polinomial y no se encuentra otra explicación que atribuirlo al entrenamiento y ajuste de parámetros.

Tabla 4.16: Métricas obtenidas con algoritmo OC-SVM con kernel sigmoidal.

Exactitud	Precisión	Sensibilidad	F1-score
0.8046	0.1591	0.9150	0.2711

El análisis de resultados del resto de turbomáquinas es bastante similar a lo expuesto anteriormente. En general los algoritmos que mejor se desempeñaron en la clasificación de los datos son los algoritmos Elliptic Envelope y Local Outlier Factor. Si bien en algunos casos particulares como con las Bombas 302 y 2211, el algoritmo Isolation Forest tuvo un rendimiento relativamente bueno (al menos en términos de Frontera de decisión), el rendimiento general ubica a este algoritmo como el de peor desempeño, en conjunto con OC-SVM. Aunque en este último caso sólo se ajustaron los parámetros *nu* y la función kernel, tal vez modificando otros parámetros ajustables como *gamma* se obtendrían mejores resultados.

Por su parte, el parámetro *contamination* puede haber influido en el mal desempeño de algunos modelos, debido a que los datos de cada turbomáquina son distintos, y en algunos casos había muy pocos o casi ningún dato "Anormal". Este aspecto debió considerarse utilizando un factor de contaminación para cada caso y el mismo para todos.

Las métricas no aportan demasiado en cuanto al real desempeño de cada modelo, de hecho, en algunos casos se tienen métricas muy similares entre un algoritmo y otro, pero al visualizar la frontera de decisión los resultados son diferentes. Como el objetivo es definir una región de operación normal para cada turbomáquina la principal herramienta de evaluación es la frontera de decisión junto al criterio experto.

Como se mencionó anteriormente se realizaron 2 versiones para el desarrollo de modelos. En la versión II se implementa el algoritmo HDBSCAN_flat que permite tanto el etiquetado de los datos, como también predecir la etiqueta de nuevos datos, asignándoles las clases identificadas durante entrenamiento del modelo. De modo que este algoritmo se usa para el etiquetado de los datos y para el monitoreo (clasificar nuevos datos en las clases normal y anormal).

Los datos son procesados aplicando los pasos de procesamiento descritos en el capítulo anterior, estos son: Normalización max-min, condición de servicio y tratamiento de datos faltantes.

Entrenamiento del modelo

Para el entrenamiento se utilizan datos de un período de 2 años (2020 y 2021) muestreados cada 10 minutos. Se usa el algoritmo HDBSCAN_flat y se ajustan los

parámetros: *n_clusters* y *min_cluster_size*, evaluando diferentes combinaciones mediante visualización del agrupamiento de datos. Los parámetros encontrados para cada turbomáquina se encuentran en los siguientes rangos.

- *n_clusters:* [2, 3]
- *min_cluster_size:* [5-15]

Nuevamente para ejemplificar el procedimiento seguido, se ilustra con los resultados de la bomba de licor verde 303.

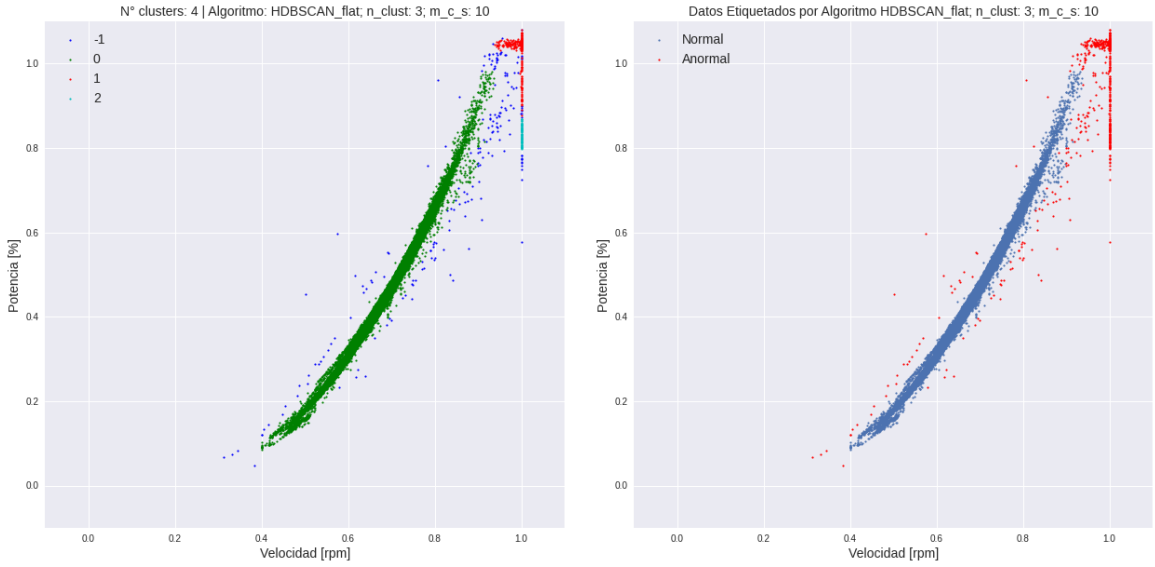


Figura 4.17: Izquierda: Resultados del agrupamiento del algoritmo HDBSCAN_flat. Derecha: re-etiquetado en clases normal y anormal.

El re-etiquetado de los datos es similar a lo realizado en la versión anterior. El criterio establece que la clase mayoritaria es la clase “normal”, aunque en algunos casos fue necesario incorporar otras clases que se consideraron parte de la operación normal del equipo.

En la etapa de entrenamiento y ajuste de parámetros se observó que la frontera de decisión definida por el algoritmo HDBSCAN_flat es muy sensible a la cantidad de ruido que tienen los datos. Cuando se tienen muchos puntos aislados alrededor del *cluster* “normal”, se observó que la frontera de decisión se ajustaba perfectamente a los datos normales, sin embargo, cuando había poco ruido, la frontera era más “permisiva” y el ajuste tenía un mayor margen.

La cantidad de ruido se podía regular cambiando la combinación de parámetros, y se optó por permitir la presencia de ruido tal que la frontera de decisión se ajustara muy bien a los datos “normales”.

Debido a lo anterior algunos modelos quedaron sobreajustados a los datos de entrenamiento, lo que en algunos casos mostró presencia de falsos positivos, como el caso del ventilador 256 mostrado en la Figura 4.18, donde se podría asegurar que las muestras positivas (color rojo) en realidad corresponderían al comportamiento normal del equipo.

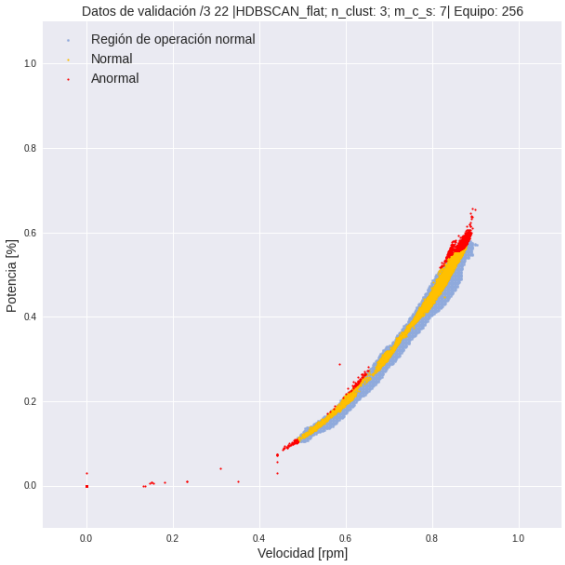


Figura 4.18: Datos de validación evaluados por modelo HDBSCAN_flat desarrollado para ventilador 256.

Es debido a esto que se recomienda que los datos de entrenamiento contengan información sobre todos los modos de operación normal del equipo, en otras palabras, los datos de Potencia y Velocidad en todo el rango de operación normal, evitando así regiones de operación normal incompletas como el caso mostrado y otros equipos como las Bombas 2202 y 1202, y los Ventiladores de tiro inducido 258 y 257, por nombrar algunos (revisar Anexo).

Por otro lado, se recomienda que los datos de entrenamiento contengan exclusivamente información sobre la operación normal del equipo sin otros modos de operación defectuosos, para lo cual es necesario conocer de antemano qué datos corresponde realmente a una operación normal del equipo.

En dos de los equipos estudiados se tiene disponible el flujo en lugar de la velocidad (Bombas 1116 y 2116). en estos casos no se observa la misma correlación no-lineal que se tiene con la velocidad y la potencia. Como se observa en la Figura 4.19, la potencia se mantiene en un rango bastante acotado para diferentes valores de flujo, sin embargo, también definen una región de operación característica.

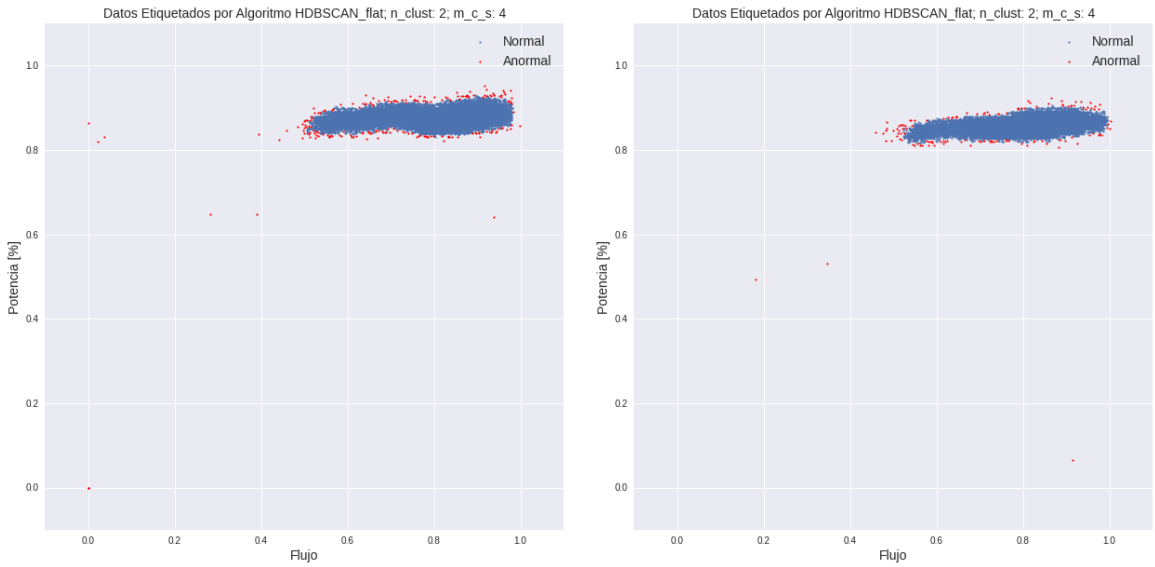


Figura 4.19: Agrupamiento realizado por algoritmo HDBSCAN_flat; Izquierda Bomba 1116; Derecha Bomba 2116.

Existen otros equipos que son operados manualmente, razón por la cual no se tiene una región continua de operación. Esto puede observarse en los datos del Soplador 1355 mostrados en la Figura 4.20, donde claramente se tiene diferentes bandas de operación a una determinada velocidad. Esto supone un problema para el agrupamiento de los datos en una sola región normal de operación debido a la discontinuidad observada.

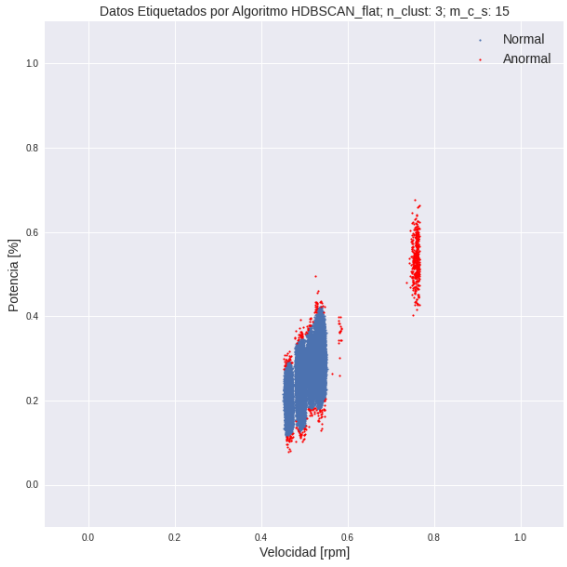


Figura 4.20: Agrupamiento realizado por algoritmo HDBSCAN_flat en Soplador 1355.

Además, debido a la cantidad de bandas, resulta difícil determinar cuáles corresponden a una operación normal. En la Figura 4.20, el *cluster* ubicado en la parte superior (color rojo), podría en efecto ser un modo de operación normal del equipo, pero debido a su lejanía con el resto fue clasificado como anormal. Situaciones como estas reafirman la necesidad de entrenar los modelos solamente con datos normales, y que idealmente incluyan todos los modos de operación normal.

Validación modelo

Para validar el modelo se utilizan dos conjuntos de datos:

- Conjunto 1: Periodo de 3 meses (12/21 al 2/22) muestreados cada 10 minutos.
- Conjunto 2: Período de 1 mes (3/22) muestreados cada 1 minuto.

En la Figura 4.21 se puede ver la clasificación realizada por el modelo HDBSCAN_flat en los dos conjuntos de validación. A la izquierda el conjunto 1, a la derecha el conjunto 2.

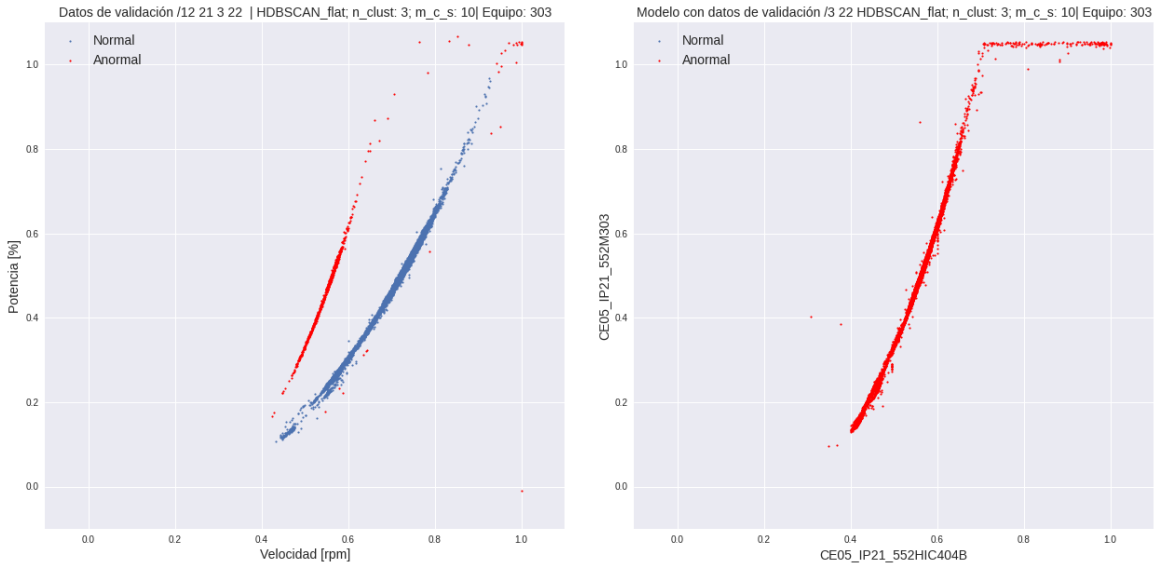


Figura 4.21: Clasificación de los datos de validación con algoritmo HDBSCAN_flat equipo Bomba 303

Evaluación modelo

Dado que las métricas no fueron una buena herramienta de evaluación en la versión anterior, se decidió evaluar el desempeño de los modelos solamente mediante visualización de la Frontera de decisión.

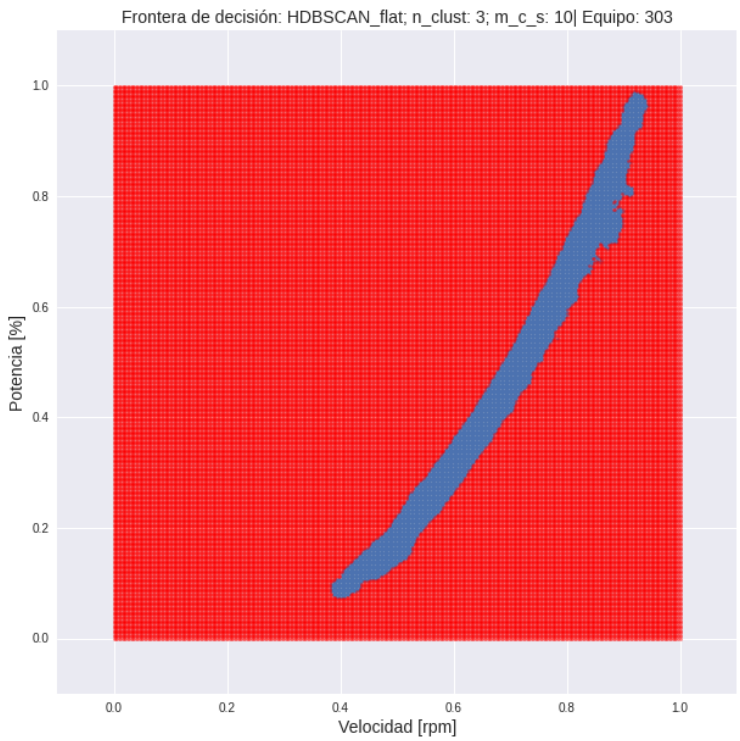


Figura 4.22: Frontera de decisión del algoritmo HDBSCAN_flat

En la Figura 4.22 se observa que la frontera de decisión se ajusta muy bien a los datos de entrenamiento, y además no tiene discontinuidades, lo cual es indicativo de un buen desempeño del modelo logrando el resultado buscado para la región de operación normal.

Implementación de modelos

Para el monitoreo de los equipos se estableció el formato de visualización mostrado en la Figura 4.23, donde en azul claro se observa la región de operación normal del equipo, en amarillo las muestras correspondientes a la operación normal y en rojo las muestras asociadas a una operación anormal.

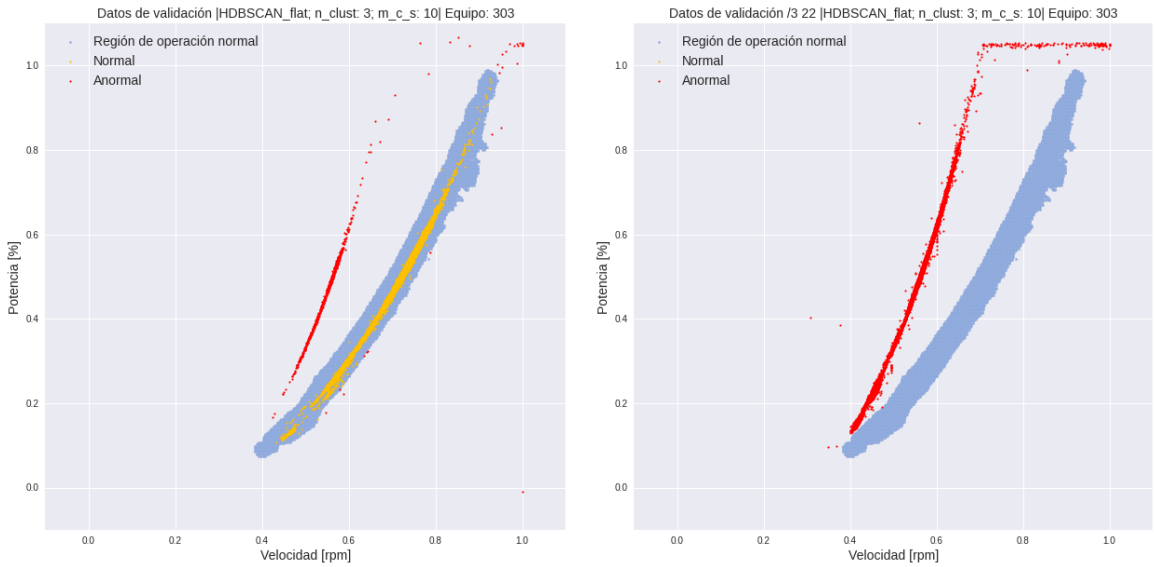


Figura 4.23: Visualización de los datos de validación junto a la región normal de operación.

En la Figura 4.23 se muestra el caso particular de la bomba 303. La imagen de la izquierda corresponde a un periodo de 3 meses donde el equipo estuvo operando una buena parte del tiempo de forma normal, sin embargo, en algún momento operó fuera de la región normal. En este esquema de visualización no se está considerando la temporalidad de los datos, no se visualiza si la operación anormal fue hacia el principio o el final del periodo. Sin embargo, la imagen de la derecha de la Figura 4.23 corresponde a un periodo posterior al de la izquierda, donde se observa que en el último mes el equipo estuvo operando totalmente fuera de su región normal de operación.

Durante una inspección de la Bomba 303 se encontró que las líneas de licor verde que llegaban al equipo estaban obstruidas como muestra la Figura 4.24, lo que estaría generando que el equipo operara con más carga de la habitual.

Figura 4.24: Líneas de licor verde obstruidas en las cercanías de la Bomba 303.

En cuanto a la implementación de los modelos para monitoreo de condición en línea, se logró llegar a una fase de prueba de concepto, donde se desarrolló la interfaz de visualización del monitoreo de los equipos turbomáquinas en la herramienta PIVision utilizando como referencia las gráficas de región de operación desarrolladas previamente en *Python*. En la Figura 4.23 se tiene la región de operación normal en azul claro, y cada 10 minutos se toma una muestra de la velocidad y la potencia del equipo que es graficada en la interfaz, permitiendo visualizar si el equipo ha estado operando en condiciones anormales cuando los puntos son graficados fuera de la región definida como de operación normal.

Durante la prueba de concepto no fue posible la ejecución *en línea* de los modelos para etiquetar automáticamente los nuevos puntos como *normales o anormales*, puesto que la plataforma y recursos necesarios para la ejecución continua de un modelo desarrollado en *Python* y la comunicación de este con la plataforma PIVision aún no han sido implementadas en Arauco. De este modo, en la etapa de prueba de concepto, es aún necesaria la supervisión de un operador o un mantenedor para advertir cuando un equipo sale de una región normal de operación que ha sido previamente estimada por los modelos desarrollados. Por tanto, la ejecución *en línea* de los modelos desarrollados para monitoreo y notificación de la operación anómala de un equipo queda propuesta como trabajo futuro.

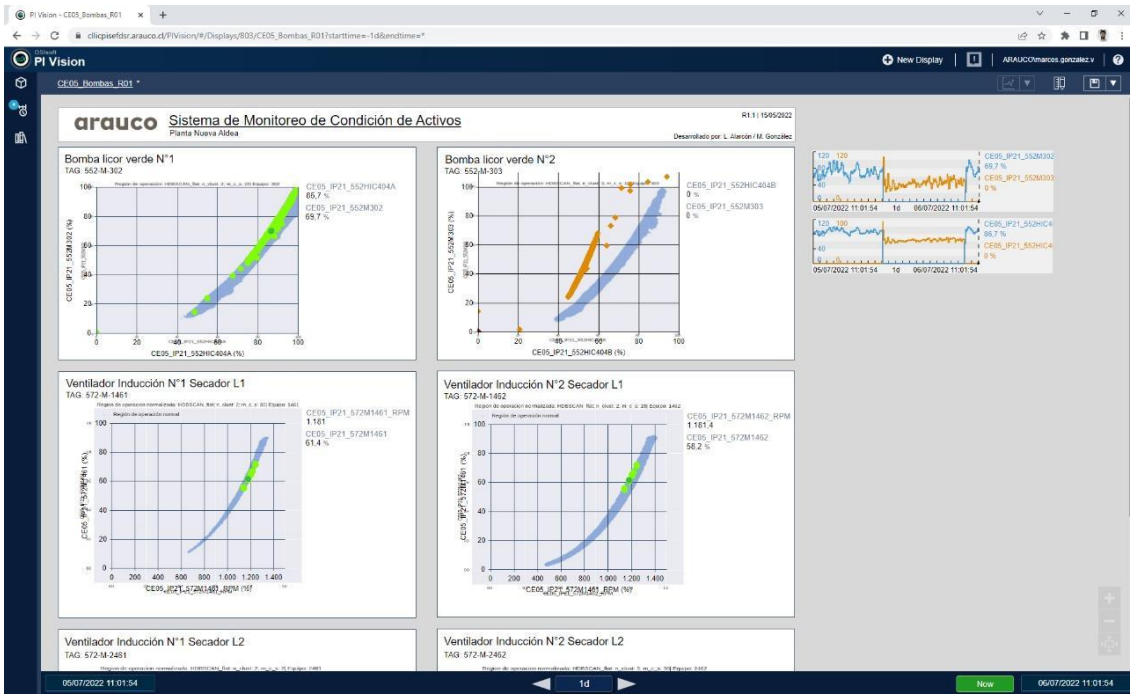


Figura 4.25: Interfaz de monitoreo desarrollada en PI Vision.

Niveles de alerta

Para buscar limitar las notificaciones de alerta, la clase "Anormal" se diferenció en 2 clases: Anormal leve y Anormal crítico. Para ello fue necesario establecer límites en la probabilidad de pertenecer a la clase normal.

Esta variante fue explorada en el caso de la Bomba de licor verde 303, estableciendo los siguientes límites.

- $Anormal\ crítico$: $P(normal) < T1 = 0.05$
- $Anormal\ leve$: $T1 = 0.05 \leq P(normal) < T2 = 0.2$

De esta manera, como se puede observar en la Figura 4.26, las muestras anormales más cercanas a la región normal son clasificados como "Anormal leve", mientras que las muestras más lejanas se clasifican como "Anormal crítico".

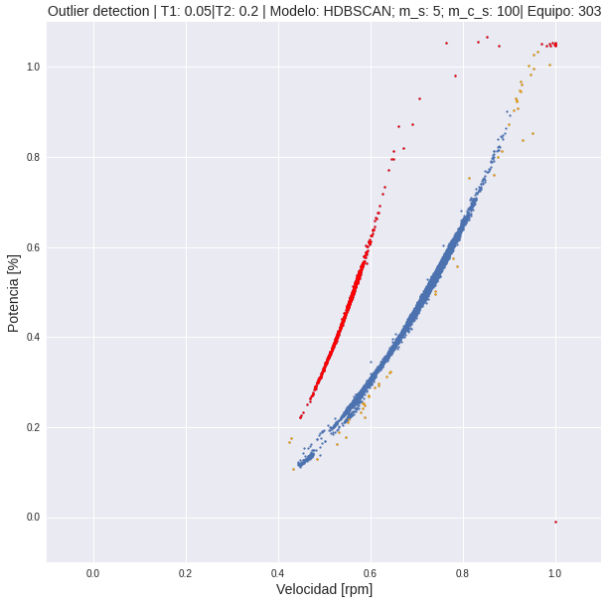


Figura 4.26: Niveles de alarma en Bomba 303.

Por otro lado, al observar la Frontera de decisión (izquierda de la Figura 4.27), notamos que la región anormal leve es casi indistinguible de la región normal y no aporta mucho margen entre las regiones normal y crítica.

Por otro lado, al reducir el límite T1 con el fin de expandir la región anormal leve (derecha de la Figura 4.27), vemos que el resultado es una región irregular que pierde la forma de la región normal.

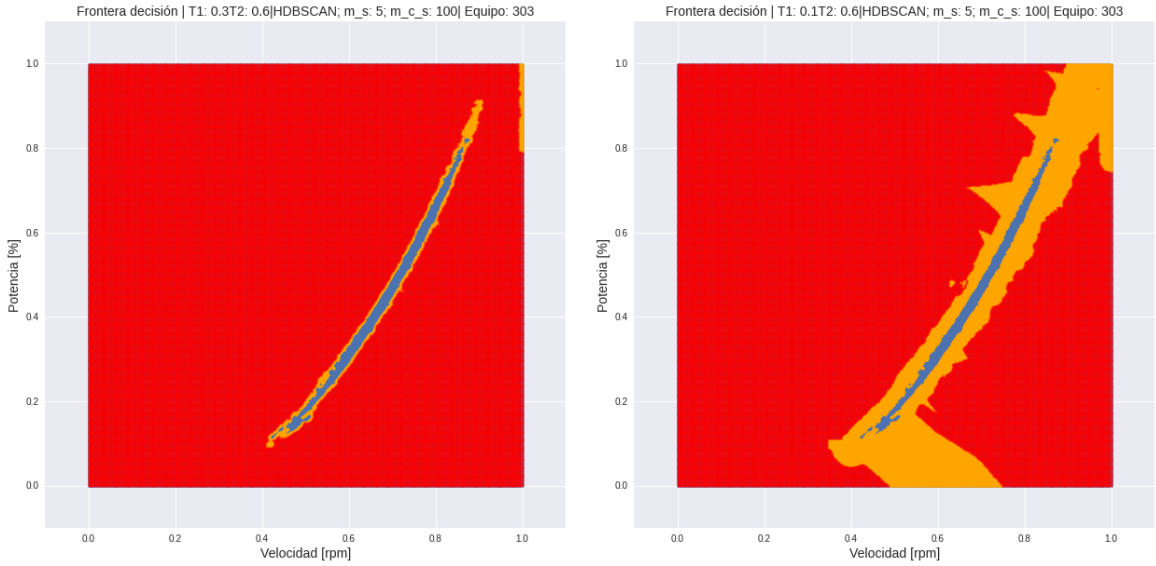


Figura 4.27: Frontera de decisión para 2 niveles de alarma. Izquierda

El resultado buscado es que la región anormal leve sea un margen entre la región de operación normal y la región anormal crítica, sin embargo, el ajuste de ese margen es muy sensible, y cuando se quiere aumentar el margen la región pierde la forma buscada.

Para llegar a estos resultados, fue necesario utilizar funciones personalizadas [33] para obtener las probabilidades de pertenecer a la clase “Normal”, sin embargo, cuando existen más de 2 *clusters* resulta bastante engorroso trabajar con estas probabilidades. Como no se obtuvieron resultados muy alentadores, esta alternativa no fue mayormente explotada.

4.3. Prensas de desplazamiento

4.3.1. Adquisición de datos

El registro de eventos en las prensas de desplazamiento contiene información sobre variados tipos de fallas en las prensas de línea 1, se decidió seleccionar el tipo de falla más recurrente y de mayor impacto en la producción: Peak de torque. Cada uno de estos eventos conlleva a la detención de la prensa, siendo en muchos casos necesario realizar el lavado con agua a alta presión y una verificación de la integridad mecánica de diversos componentes del equipo antes de poder ser puesto nuevamente en servicio. En Tabla 4.17 se resume la cantidad de eventos registrados para cada tipo de falla.

Tabla 4.17: Tipos de fallas registradas en prensas de línea 1

Tipo de Falla	Eventos registrados
Alta carga tornillo desmenuzador	12
Peak de torque	1371
Peak en indicación de presión de alimentación	61
Taco de partida	145
Taco en tornillo de alimentación	4
Taco en tornillo desmenuzador	19
Peak de presión en indicación de batea	121
Otro	90

Los eventos registrados y expuestos en la Tabla 4.17 consideran la totalidad de las prensas de Línea 1, la Tabla 4.18 por su parte desglosa los eventos de alto torque registrados para cada tipo de prensa

Tabla 4.18: Eventos de alto torque registrados en prensas línea 1.

Tipo de prensa	Eventos registrados	Variables disponibles
Pre-Oxígeno #1	2	20
Pre-Oxígeno #2	59	18
Post-Oxígeno	39	18
Pre-Blanqueo	209	13
D0	247	17
EoP	110	17
D1	246	16
D2	451	19

Antes de extraer los datos fue necesario entender el funcionamiento de las prensas de desplazamiento, la función y ubicación de cada una en el proceso productivo, para lo cual se utilizó documentación técnica de estos equipos y de los procesos de Lavado, Deslignificación y Blanqueo Gracias a esto fue posible tener una mejor idea de qué datos y variables podían ser relevantes para estudiar el fenómeno de altos torques.

Los datos de monitoreo corresponden a variables de operación en el entorno del equipo. Para identificar las variables más relevantes y sus respectivos códigos se utilizó como apoyo imágenes del Sistema de Control Distribuido (DCS) como la que se puede ver en la Figura 4.28.

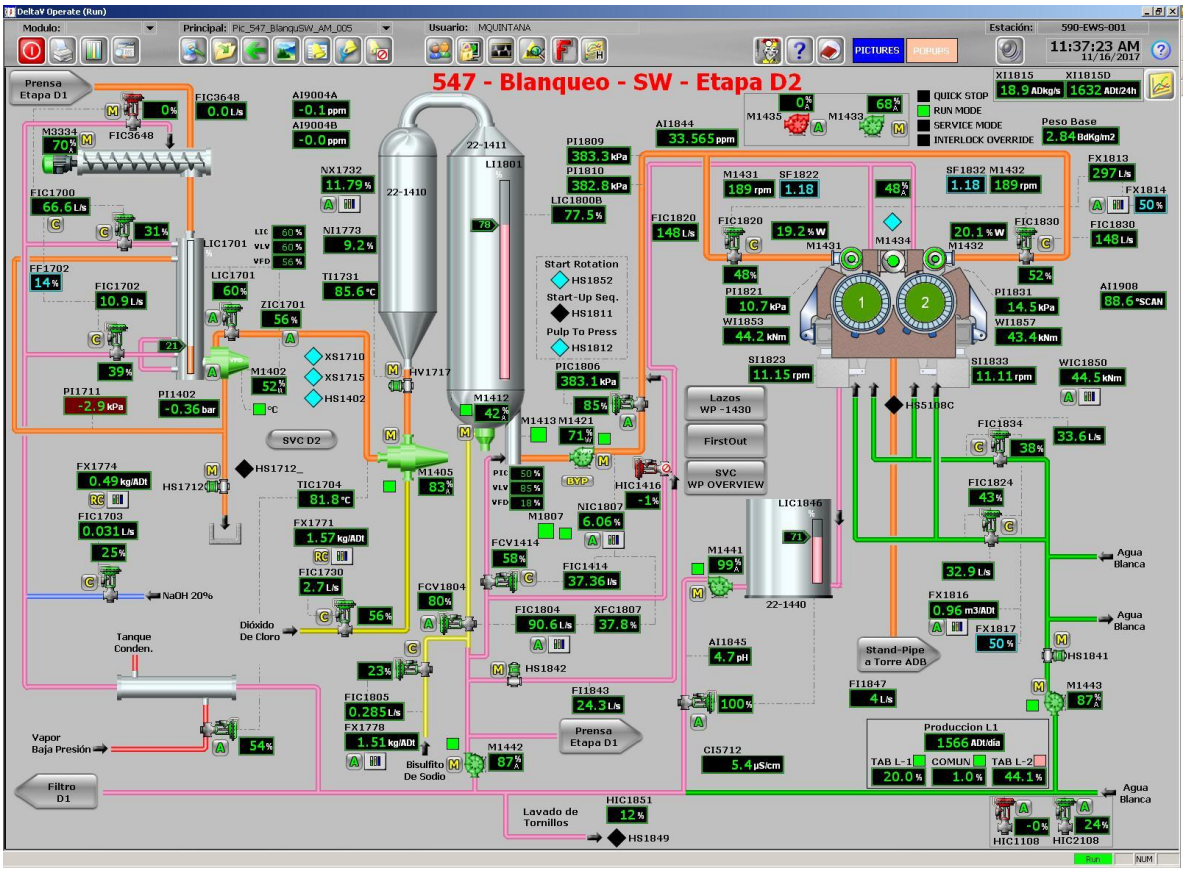


Figura 4.28: Pantalla del Sistema de Control Distribuido para la etapa D2 del área de blanqueo.

Con los *tags* de cada variable, se verificó su existencia en la base de datos IP21 mediante el software *Wedge*. Con esta herramienta también se examina la calidad de los datos mediante la visualización de las variables de monitoreo en el periodo de interés, verificando principalmente que las mediciones tengan sentido y se muevan en los rangos correctos.

De esta manera se levantó información de todas las prensas de Línea 1 y de sus variables respectivas. En este proceso se encontró que muchas de las variables no existían en la plataforma *Wedge* mientras que otras variables sí existían, pero las mediciones estaban fijas en un valor indicando que el sensor no estaba funcionando correctamente. La última columna de la Tabla 4.18 muestra la cantidad de variables disponibles para cada prensa.

En virtud de lo anterior fue seleccionada la prensa de desplazamiento D2 ya que la información disponible es más completa en comparación con el resto de las prensas. La Tabla 4.19 muestra el detalle de las variables de monitoreo disponibles para la prensa seleccionada.

Tabla 4.19: Variables de monitoreo disponibles prensa D2 línea 1.

Variable	Tag	Unidad
Torque motor hidráulico 1	CE05_IP21_547WI1853	KNm
Torque motor hidráulico 2	CE05_IP21_547WI1857	KNm
Torque máximo prensa D2	CE05_IP21_547WIC1850	KNm
Flujo 1 pulpa a prensa D2	CE05_IP21_547FIC1820	l/s
Flujo 2 pulpa a prensa D2	CE05_IP21_547FIC1830	l/s
Flujo total pulpa a prensa D2	CE05_IP21_547FX1813	l/s
Presión pulpa entrada 1	CE05_IP21_547PI1821	kPa
Presión pulpa entrada 2	CE05_IP21_547PI1831	kPa
Consistencia pulpa entrada	CE05_IP21_547NIC1807	% (P/P)
Consistencia pulpa salida	CE05_IP21_547FX1816CONS	% (P/P)
Flujo agua de lavado 1	CE05_IP21_547FIC1824	m^3
Flujo agua de lavado 2	CE05_IP21_547FIC1834	m^3
Velocidad rodillo 1	CE05_IP21_547SI1823	rpm
Velocidad rodillo 2	CE05_IP21_547SI1833	rpm
Carga tornillo descarga	CE05_IP21_547M1434	%A
Carga tornillo alimentación 1	CE05_IP21_547M1431	%A
Carga tornillo alimentación 2	CE05_IP21_547M1432	%A
Velocidad tornillo alimentación 1	CE05_IP21_547M1431_RPM	rpm
Velocidad tornillo alimentación 2	CE05_IP21_547M1432_RPM	rpm

Los eventos registrados abarcan un periodo de 3 años y las señales de monitoreo están muestreadas cada 10 segundos. Los datos fueron exportados utilizando la plataforma *Wedge* en archivos formato *excel*.

4.3.2. Caracterización del fenómeno

El fenómeno de alto torque corresponde a un aumento repentino de la presión hidráulica de los motores hidráulicos que accionan los rodillos. El equipo está configurado para detenerse en caso de que la presión supere un cierto umbral como medida de seguridad.

Para verificar si la falla por alto torque es observable en las señales de monitoreo se procedió a visualizar las variables en periodos donde existieron fallas registradas, tal como muestra la Figura 4.19.

Al observar el comportamiento de las variables, se puede encontró que existe un peak no solo en la señal de torque, sino también en la velocidad en los rodillos, la carga de los tornillos de alimentación y la consistencia de la pulpa a la entrada de la prensa.

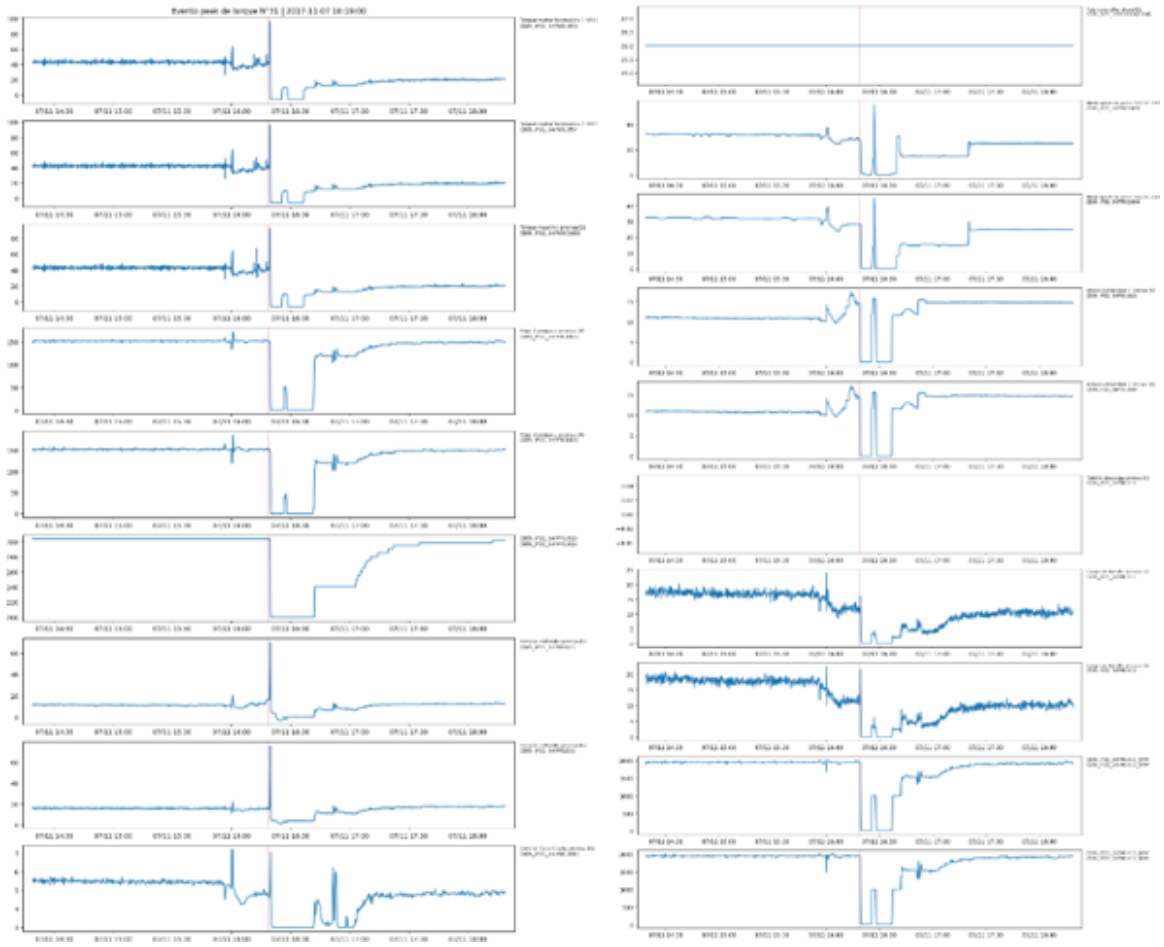


Figura 4.29: Visualización de variables de monitoreo para un evento en particular.

Las otras variables caen a cero cuando el equipo se detiene mientras que la variable *Carga tornillo descarga* no tiene mediciones y por tanto esta variable fue eliminada. El tipo de falla por alto torque es en efecto visualizable en las variables monitoreadas, lo cual es buen indicio para el desarrollo de los modelos predictivos. Sin embargo, a simple vista no se observan indicadores o precursores del evento en ninguna de las variables para la mayoría de los eventos registrados.

En ciertos casos, se observan peaks de torque antes de que ocurra el evento registrado, sin embargo, a pesar de tener magnitudes similares al evento en sí, la prensa sigue operando. En otros casos ni siquiera se observa un peak claro en el evento registrado. Esto podría deberse a que el peak simplemente no fue captado (recordemos que la frecuencia de muestreo es de una muestra cada 10 segundos).

En algunos casos particulares, el torque tiene un aumento progresivo y no repentino. En estos casos se observa un comportamiento diferente en la velocidad de los rodillos, ya que tiene un aumento repentino y se mantiene en este valor (~20 rpm) hasta que el torque supera el umbral y el equipo se detiene. Cabe destacar que el valor de la consistencia de pulpa a la entrada permanece prácticamente constante en todo momento hasta que ocurre la detención por alto torque. Este comportamiento tiene sentido en parte ya que el torque está relacionado con la velocidad. Si la velocidad de los rodillos aumenta, por ejemplo, el torque resistente aumentará.

El control del sistema también es un factor relevante. Dependiendo de cuál variable está siendo controlada, se tendrán diferentes comportamientos: Si se controla el torque, este debería mantenerse más o menos constante, y en consecuencia la velocidad de los rodillos cambiará de forma inversa con el flujo másico de pulpa que ingresa a la prensa. Por otro lado, si se controla la velocidad, el torque ejercido por los motores hidráulicos deberá aumentar junto con el aumento en el flujo de aceite para mantener una velocidad constante de operación.

4.3.3. Pre-procesamiento de los datos

Durante la visualización se encontró que algunos eventos fueron mal registrados en la bitácora de eventos de eventos de la prensa, por un lado, debido a errores en el formato de la fecha algunos eventos no podían ser leídos por el código, de modo que bastó con corregir el formato de la fecha para que pudieran ser correctamente leídos.

Por otro lado, al visualizar las variables en algunas fechas registradas no se encontró ninguna anormalidad en las señales, de manera que fue necesario ampliar la ventana temporal para verificar si efectivamente existía un evento de alto torque en el entorno de la fecha

registrada. Se encontró que en la mayoría de los casos la fecha registrada tenía cerca de un día de desfase.

Una vez corregidas las fechas se procedió a seleccionar los eventos. Para ello se definieron dos categorías:

Eventos aislados

Cuando la prensa se detiene por alto torque el procedimiento que se sigue es abrir la batea de la prensa, limpiar los rodillos y volver a su ponerla en operación, sin embargo, como no se tiene claro el motivo del alto torque, muchas veces las condiciones de operación siguen siendo las mismas que antes y la prensa vuelva a sufrir un evento de alto torque quedando nuevamente fuera de servicio. Este patrón se repite en varios de los eventos registrados y se observa en la señal como una seguidilla de peaks de torque tal como muestra la Figura 4.30.

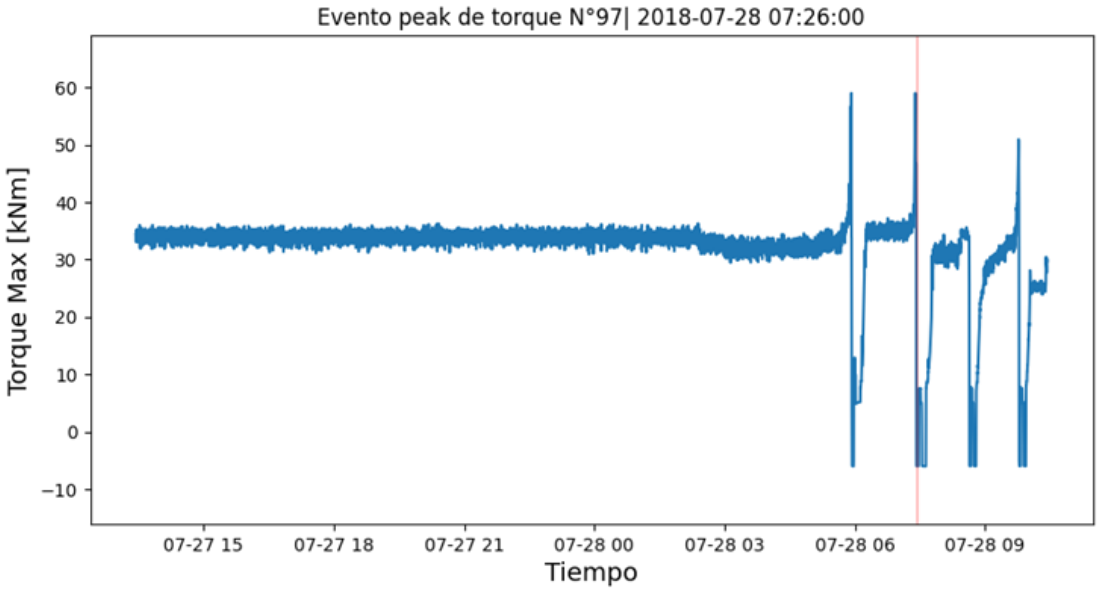


Figura 4.30: Seguidilla de peaks de torque. La línea roja señala un evento no-aislado.

En este contexto sólo interesa predecir el primer peak de torque que desencadena los posteriores, ya que los siguientes peaks serían una consecuencia de las mismas condiciones que explican el primer evento. De esta forma se definió no considerar en el estudio los eventos repetitivos posterior a un evento de alto torque inicial.

El criterio consiste en conservar solamente aquellos eventos tales que en horas previas no existan otros eventos. En definitiva, lo que importa es que no existan otros eventos durante una determinada cantidad de horas antes de que ocurra el evento en cuestión.

Para definir la cantidad de horas en que se observa si hay o no otro evento, se graficó la cantidad de eventos aislados en función de las horas observadas antes del evento, obteniendo la curva de la Figura 4.31.

Figura 4.31: Cantidad de eventos aislados en función de las horas observadas.

Considerando que el tiempo de residencia promedio de la pulpa en el digestor de astillas (formación de la pulpa) es de 15 horas, se podría decir que las propiedades de la pulpa son más o menos estables en ese orden de tiempo. Además, se busca un punto donde la cantidad de eventos no varíe demasiado ante el cambio en cantidad de horas observadas.

De esta forma se consideró un periodo de 18 horas de observación antes del evento, para establecer si corresponde a un evento aislado o no. Con este criterio de selección la cantidad de eventos se redujo de 451 a 148.

Estabilidad operacional previa a los eventos

Las prensas sufren detenciones por otros motivos u otros tipos de falla (Tabla 4.17), por tanto, se definió otro criterio para no considerar aquellos eventos asociados a una

operación inestable, esto es, que tuvieran periodos fuera de servicio en las 18 horas previas al evento de alto torque, tal como se puede observar en la Figura 4.32.

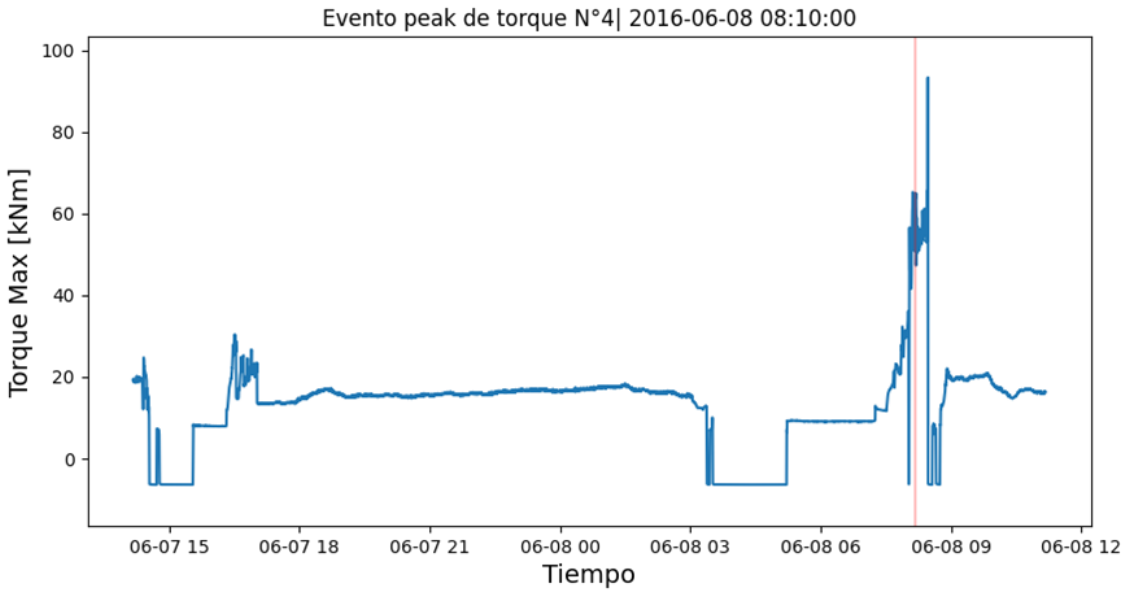


Figura 4.32: Evento en operación inestable. El peak torque está señalado por la línea roja.

Cabe destacar que también se eliminaron eventos que no eran realmente eventos, es decir, si bien existía un peak de torque, la prensa seguía operando luego del peak. También se eliminaron eventos cuyas mediciones eran incorrectas, en particular eventos en que las señales se mantienen en un valor fijo. De esta manera de los 148 eventos bajo operación estable, se conservaron 81 eventos.

Estos 81 eventos aislados y ocurridos dentro de un periodo de operación estable del equipo son utilizados para el entrenamiento de los modelos predictivos. La estructura de los datos corresponde a ventanas temporales como la mostrada en la Figura 4.33, la cual considera 18 horas previas al evento y 2 horas posteriores.

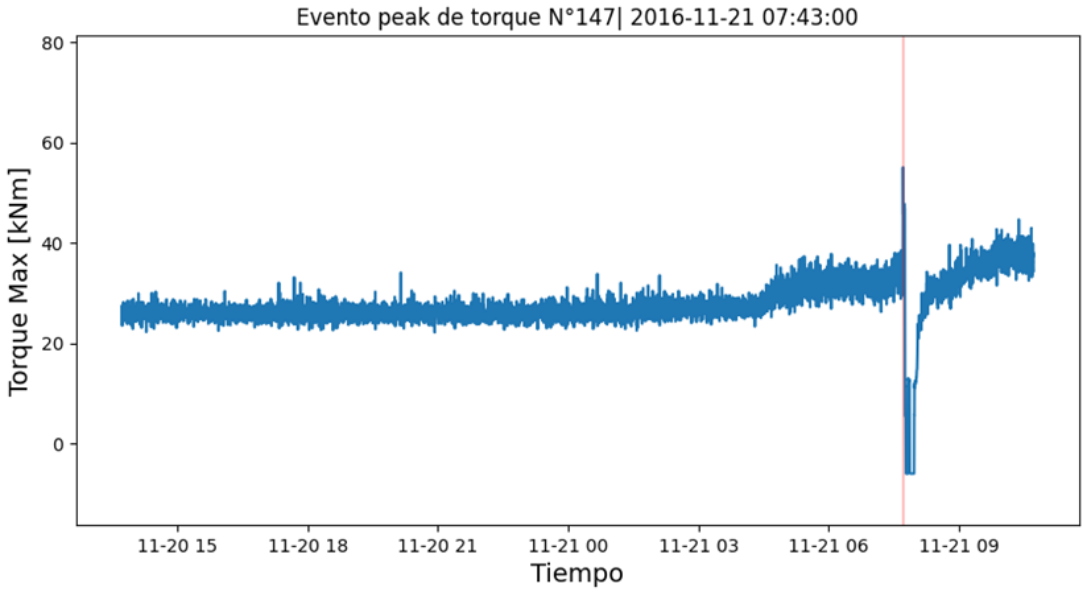


Figura 4.33: Evento aislado y estable.

Finalmente, fue necesario corregir nuevamente las fechas de los eventos para precisar el momento exacto en que ocurre el peak de torque, en otras palabras, el valor máximo inmediatamente antes de la detención de la prensa.

4.3.4. Planteamiento del problema y selección de modelos

Con el conocimiento de los datos e información disponible se procedió a definir más en detalle el problema a resolver. Para ello se plantearon diferentes propuestas a resolver, las cuales pasan a ser descritas a continuación

i) Modelos de detección de anomalías

Estos modelos son semi-supervisados, de modo que no es necesario etiquetar los datos, aunque sí debe tener conocimiento de cuáles datos corresponden a operación normal de la prensa y cuáles no.

Para entrenar modelos de detección de anomalías se utilizan sólo datos de operación normal, es decir, las primeras horas de operación de las ventanas temporales extraídas para cada evento. Luego el modelo se valida y evalúa con las señales completas, incluyendo los eventos de alto torque.

El modelo debería ser capaz de identificar cualquier comportamiento que se aleje del normal, y en consecuencia identificar precursores al evento. Si estos precursores existen, este tipo de modelo sería capaz de predecir con anticipación los eventos de alto torque, pero si las variables no exhiben comportamientos erráticos antes del evento (precursores), el modelo no aportaría capacidad predictiva y sólo reconocería el peak de torque.

Se plantea usar como modelo base un algoritmo estadístico como *Elliptic Envelope*, y luego desarrollar un modelo más complejo como *Autoencoders* y comparar los resultados de ambos.

ii) Modelos de clasificación basados en árboles

Estos modelos son supervisados por lo tanto es necesario etiquetar los datos. Para cada ventana temporal, las muestras inmediatamente anteriores al evento pueden ser etiquetadas como parte del evento asumiendo que corresponden a precursores. Las muestras previas a los precursores son etiquetadas como operación normal.

Los modelos basados en árboles permiten además identificar qué variables son más relevantes en la predicción de la etiqueta de cada muestra.

iii) Modelos de detección temprana de eventos raros

Similar a la propuesta anterior se plantea utilizar modelos de clasificación binaria para la detección temprana o predicción de eventos de alto torque, pero con algunas modificaciones clave.

En primer lugar, los eventos de alto torque sólo ocurren de manera puntual y las prensas por lo general están en operación normal la mayor parte del tiempo. Incluso en las ventanas temporales extraídas se tienen datos correspondientes a horas operación normal, mientras que cada evento ocurre en el orden de los minutos. De esta manera existe un claro desbalance en las clases normal y anormal. Existen métodos para combatir estos problemas de clasificación desbalanceados.

En segundo lugar, el orden temporal de los datos importa cuando se trata de series de tiempo ya que pueden existir patrones que se repiten en el tiempo y sólo pueden ser reconocidos al conservar ese orden temporal. Los modelos propuestos anteriormente no consideran la temporalidad de los datos, sino que toman cada muestra de forma independiente.

Se plantea usar como modelo base *redes neuronales multicapa (MLP)* con distintos métodos que apuntan a resolver el problema de clases desbalanceadas, y posteriormente desarrollar modelos que consideran patrones temporales, las *redes neuronales recurrentes*

(RNN) y *redes neuronales convolucionales (CNN)*. Este tipo de metodología se muestra en [22] como una alternativa para el estudio de problemas similares al planteado en este trabajo, en un contexto de estudio de variables continuas, entre otras, a procesos industriales (por ejemplo, predicción de cortes de hoja en una fábrica de papel).

Finalmente se optó por estudiar la tercera propuesta ya que considera aspectos que las otras no consideran y parece ser la opción más adecuada para el problema que se quiere resolver.

4.3.5. Procesamiento de datos

A continuación, se describen las distintas acciones de procesamiento realizadas con el fin de preparar los datos para el entrenamiento de los modelos.

Las distintas versiones de los modelos desarrollados requieren distintos procedimientos para el tratamiento de datos, motivo por el cual las acciones realizadas se describen de forma individual y sin un orden particular. Más adelante se detallará el procedimiento utilizado para cada versión de modelo desarrollado.

Reestructuración de datos

Las ventanas temporales que contienen a cada evento de alto torque son recortadas para reducir la cantidad de datos y sólo conservar información relevante, para ello se realizó lo siguiente.

- Los datos de monitoreo posteriores al evento no entregan información útil para la detección del alto torque por lo tanto son eliminados.
- Para reducir la cantidad de datos y disminuir los efectos del desbalance de clases, sólo se conservan las muestras correspondientes a las 8 horas previas al evento.

De esta manera las ventanas temporales de cada evento pasan de un tamaño de 20 horas a solo 8 horas como puede verse en la Figura 4.34.

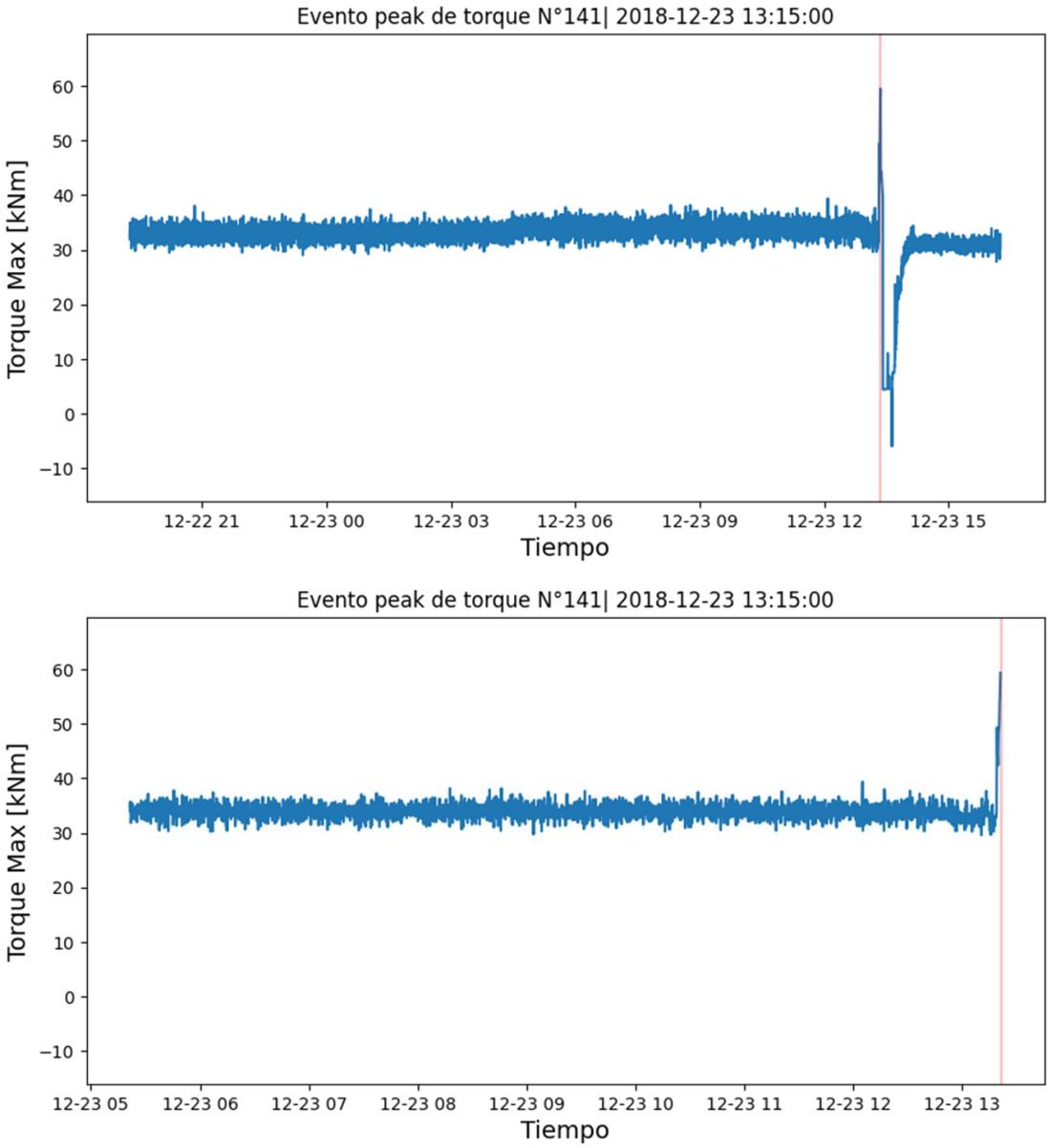


Figura 4.34: Arriba: Evento antes de ser recortado; Abajo: Evento recortado

Además, se reducen las dimensiones de los datos, pasando de un tensor 3D de la forma (tiempo, variables, eventos), a uno 2D de dimensiones (tiempo, variables). Para ello las ventanas temporales se pegan una detrás de otra formando una señal única multivariable. La Figura 4.35 ejemplifica esto con la variable de torque máximo.

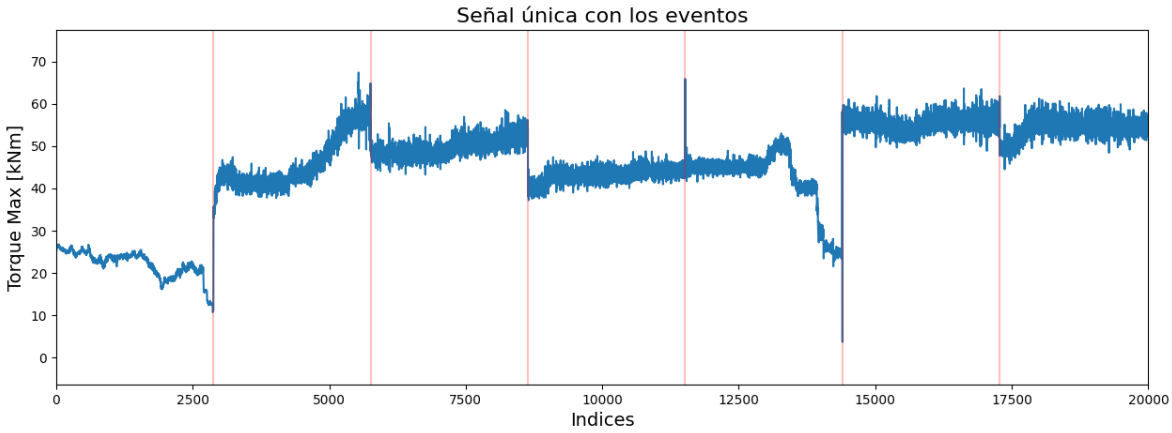


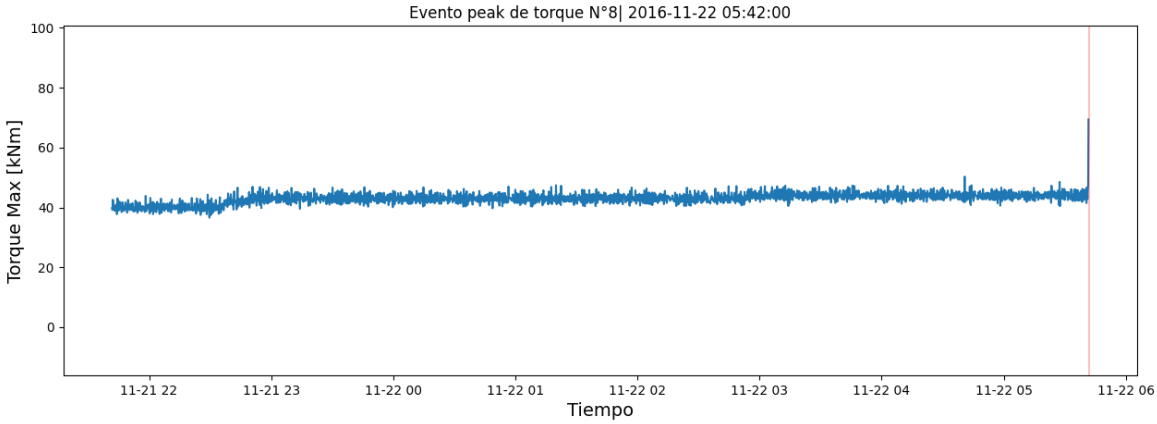
Figura 4.35: Algunas ventanas temporales convertidas en una señal única.

Etiquetado de datos

Como se trata de un problema de clasificación binaria existen dos clases de datos. Los datos de operación normal son etiquetados como Clase 0 (negativo), mientras que el evento de alto torque es la Clase 1 (positivo). Como se mencionó anteriormente, los datos inmediatamente anteriores al evento, los precursores, también son etiquetados como Clase 1.

Un detalle importante es definir cuánto tiempo antes del evento se etiqueta positivo, ya que a priori ni siquiera se sabe si existen precursores. Sin embargo, para que el modelo aporte utilidad, debe ser capaz de detectar con anticipación suficiente para dar tiempo a los operadores del equipo para tomar acción y prevenir la detención.

Por lo tanto, se consideró cómo tiempo mínimo de detección 10 minutos previos al evento. Este fue el valor inicial que se asumió para los primeros modelos desarrollados. Posteriormente se hicieron pruebas modificando este parámetro buscando mejorar la capacidad predictiva de los modelos.



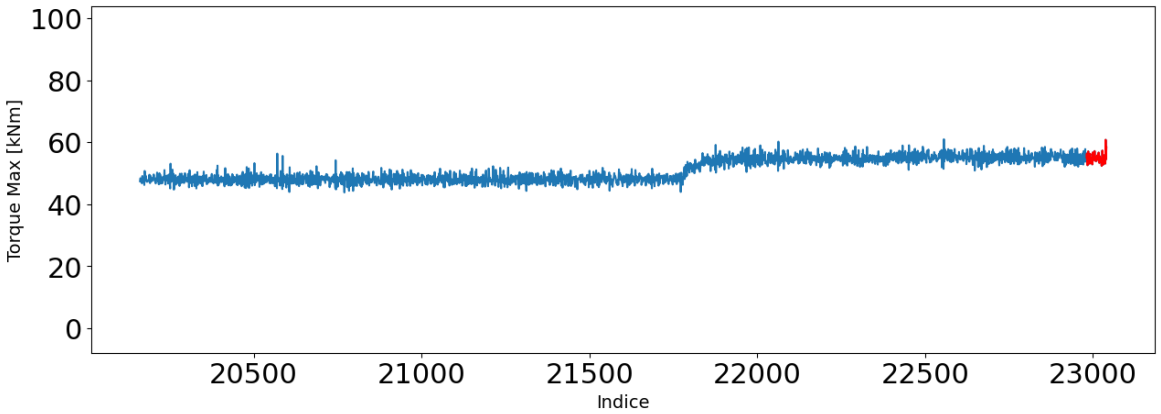


Figura 4.36: Arriba: Evento antes de etiquetar; Abajo: Evento etiquetado, en rojo los puntos etiquetados positivos.

Otro detalle importante que se ve claramente en la Figura 4.36 es que también se elimina el peak de torque, ya que genera ruido en los datos haciendo menos eficiente el aprendizaje de los modelos. En la práctica, tampoco es relevante detectar un peak de torque al mismo instante en que ocurre, pues no permite a los operadores o mantenedores tomar acción oportuna para evitar anticipadamente la detención de la prensa.

Datos faltantes

Otro paso en el procesamiento de los datos es verificar los datos faltantes para cada una de las variables monitoreadas. Se establece como criterio que, si una variable tiene más de un 10% de valores faltantes, se procede a eliminar dicha variable. Por otro lado, si no se supera ese umbral los datos faltantes se rellenan mediante interpolación lineal.

En base a lo anterior la variable *Carga tornillo descarga"* no cumplió con el criterio establecido y fue descartada. Por otro lado, no fue necesario interpolar los datos en el resto de las variables.

Temporalización

Este proceso es necesario para los modelos recurrentes y convolucionales. Consiste en la extracción de ventanas temporales de un determinado tamaño en las cuales los modelos buscan patrones temporales. El tamaño de la ventana será un parámetro relevante para el entrenamiento de dichos modelos.

El proceso se puede ver ejemplificado en la Figura 4.37, donde para cada muestra se extrae una ventana de tamaño *Lookback* que incluye la muestra misma y las que la anteceden. La etiqueta de cada ventana se corresponde con la etiqueta de la última muestra de la ventana.

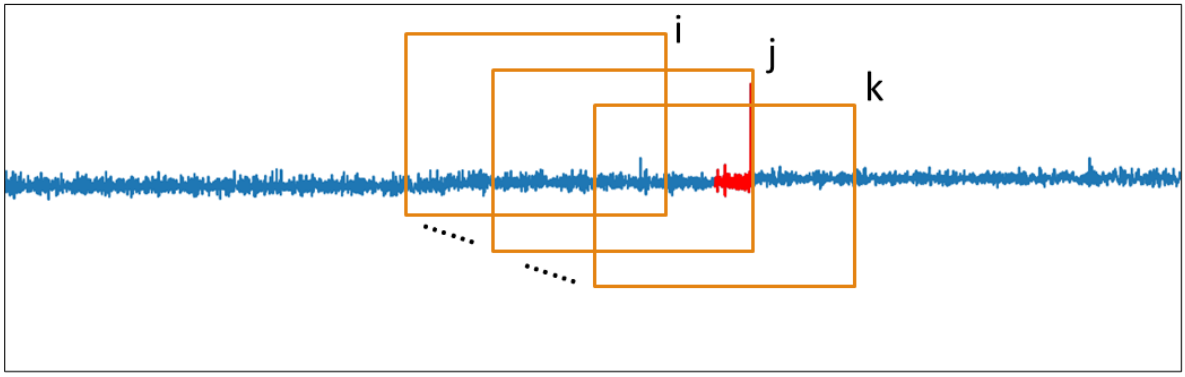


Figura 4.37: Esquema de la temporalización de dato

En la Figura 4.37 las ventanas ***i*** y ***k*** son etiquetadas negativas, mientras que la ventana ***j*** se etiqueta positiva.

Disminuir frecuencia de muestreo

En algunos casos, en particular para los modelos recurrentes y convolucionales, donde se deben temporalizar los datos, los recursos computacionales son insuficientes para almacenar tantas ventanas. Para estos casos se resolvió disminuir la frecuencia de muestreo, pasando de 1/10 [muestras/segundos], a 1 [muestra/minuto], reduciendo en efecto la cantidad de datos y en consecuencia la cantidad de ventanas generadas en la temporalización.

Subconjuntos de datos

Los datos, ya sea en forma de muestras individuales o muestras temporalizadas (ventanas temporales), son divididos de forma aleatoria en la siguiente proporción:

- Conjuntos de Entrenamiento y Validación: 80%
  - Conjunto de Entrenamiento: 64%
  - Conjunto de Validación: 16%
- Conjunto de Testeo: 20%

Normalización de datos

Es importante normalizar los datos, sobre todo cuando se entrenan redes neuronales, ya que las variables o parámetros con valores de mayor orden de magnitud van a tener una mayor influencia en los resultados de la capa de salida.

Por esto se busca que todas las variables tengan el mismo orden de magnitud, de modo que los datos son normalizados por estandarización mediante la ecuación 4.2.

$$X_{norm} = \frac{X - \underline{X_{train}}}{\sigma_{train}} \quad (4.2)$$

Cabe destacar que todos los datos son normalizados a partir de los datos del conjunto de entrenamiento.

4.3.6. Modelos base: MLP

Como modelo base para la detección temprana de eventos de alto torque se proponen las redes neuronales *Multi-Layer Perceptron*. El procesamiento de los datos para los modelos base fue el siguiente:

1. Reestructuración de datos en una única señal
2. Etiquetado de los datos
3. Dividir los datos aleatoriamente en conjuntos de Entrenamiento, Validación y Testeo.
4. Normalizar los datos a partir de los datos de Entrenamiento.

Iteración I

Como primera iteración se etiquetan positivos los 10 minutos previos al evento. En la Tabla 4.20 se especifican las diferentes variaciones de esta primera iteración. Estas variaciones tienen una arquitectura similar a la mostrada en la Figura 4.38.

Tabla 4.20:Modelos entrenados en la Iteración I.

Caso	Activación	Dropout	Método balance de clases
I	relu	0.0	NO
II	relu	0.2	NO
III	relu	0.5	NO
IV	tanh	0.0	NO
V	relu	0.0	Balance pesos
VI	relu	0.2	Balance pesos
VII	relu	0.0	SMOTE
VIII	relu	0.2	SMOTE
IX	selu	0.1 (Alpha Dropout)	NO

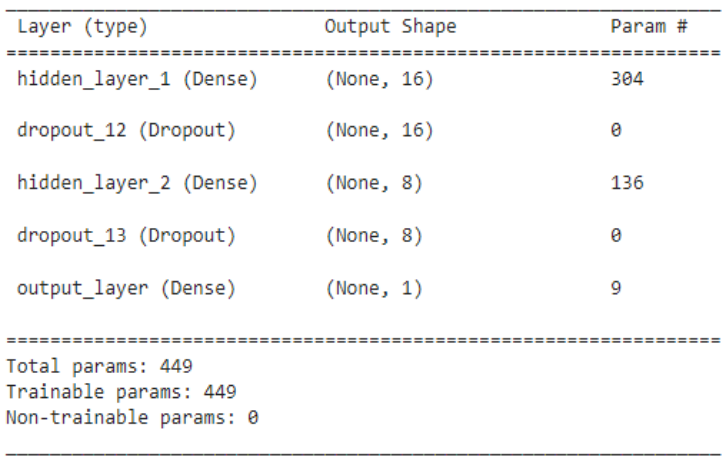


Figura 4.38: Arquitectura modelos base.

Como muestra la Figura 4.38, las características de los modelos son:

- Capa de entrada: Tiene tantas unidades como parámetros (18)
- Capas ocultas: 2 capas densas de 16 y 8 unidades respectivamente.
- Capa de salida: Una sola unidad con activación Sigmoide.

Para el entrenamiento se utilizan los siguientes parámetros:

- Tamaño de batch: 128
- Épocas: 150
- Optimizador: Adam
- Learning rate: 0.001
- Función de pérdida: Binary Cross Entropy

Caso I

Los resultados de cada caso son presentados en el formato de la Figura 4.39, donde los cuatro gráficos corresponden a las curvas de aprendizaje de las métricas calculadas en cada época de entrenamiento, destacando que la línea punteada hace referencia a las métricas calculadas en el conjunto de entrenamiento y la línea continua a las métricas calculadas en el conjunto de validación.

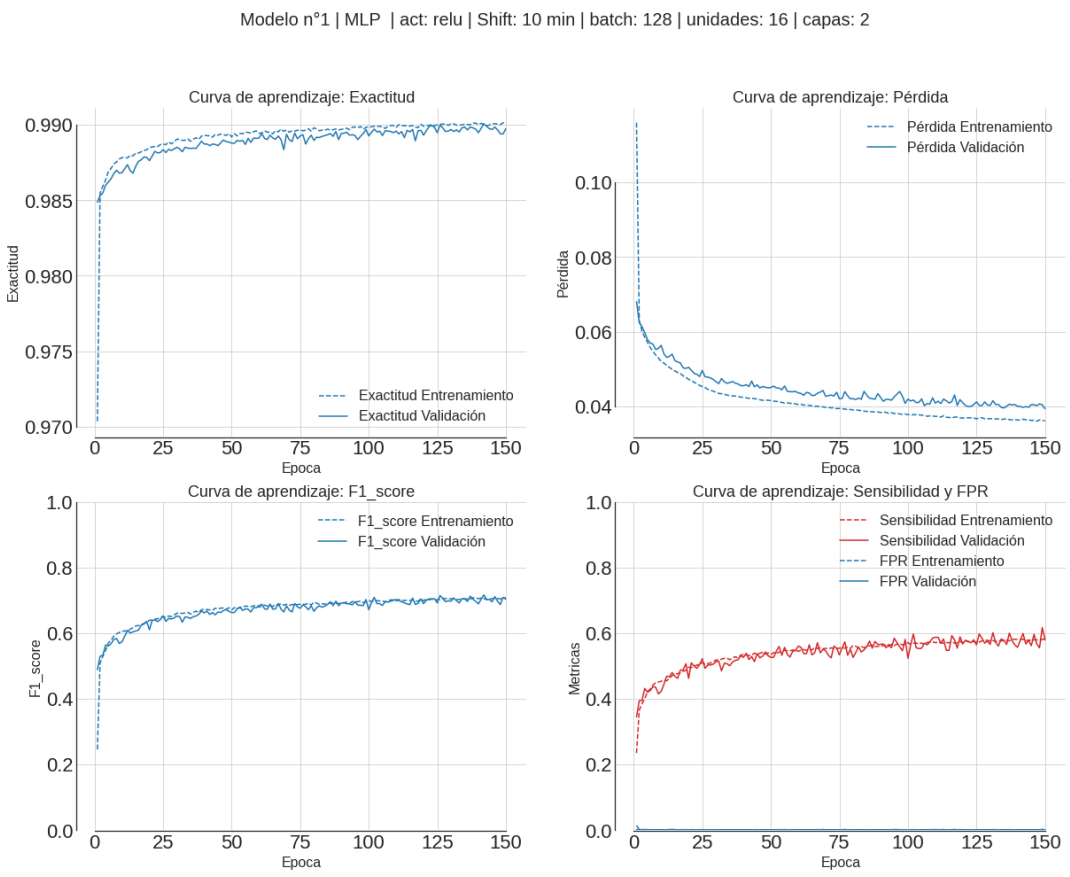


Figura 4.39: Curvas de aprendizaje caso I MLP

En la esquina superior izquierda de la Figura 4.39 se tiene la evolución de la exactitud promedio, donde se observa que en ambos casos el valor de la exactitud tiende a un valor del 99%. Sin embargo, en casos de clases desbalanceadas la exactitud no es un buen indicador del buen rendimiento del modelo.

En la esquina superior derecha, la curva de pérdida muestra un comportamiento decreciente tanto en el conjunto de entrenamiento como el de validación, lo cual es un buen indicador de la convergencia del modelo y de que el modelo no está sobre ajustado. Aunque se debe mencionar que la pérdida de validación se empieza a alejar levemente de le pérdida de entrenamiento.

La esquina inferior izquierda muestra la curva del F1-score, que llega a un valor cercano a 0.7, mientras que la sensibilidad llega a 0.6 como se ve en el gráfico de la esquina inferior derecha. Cabe destacar que la cantidad de falsos positivos se mantiene en valores muy cercanos a 0

Caso II

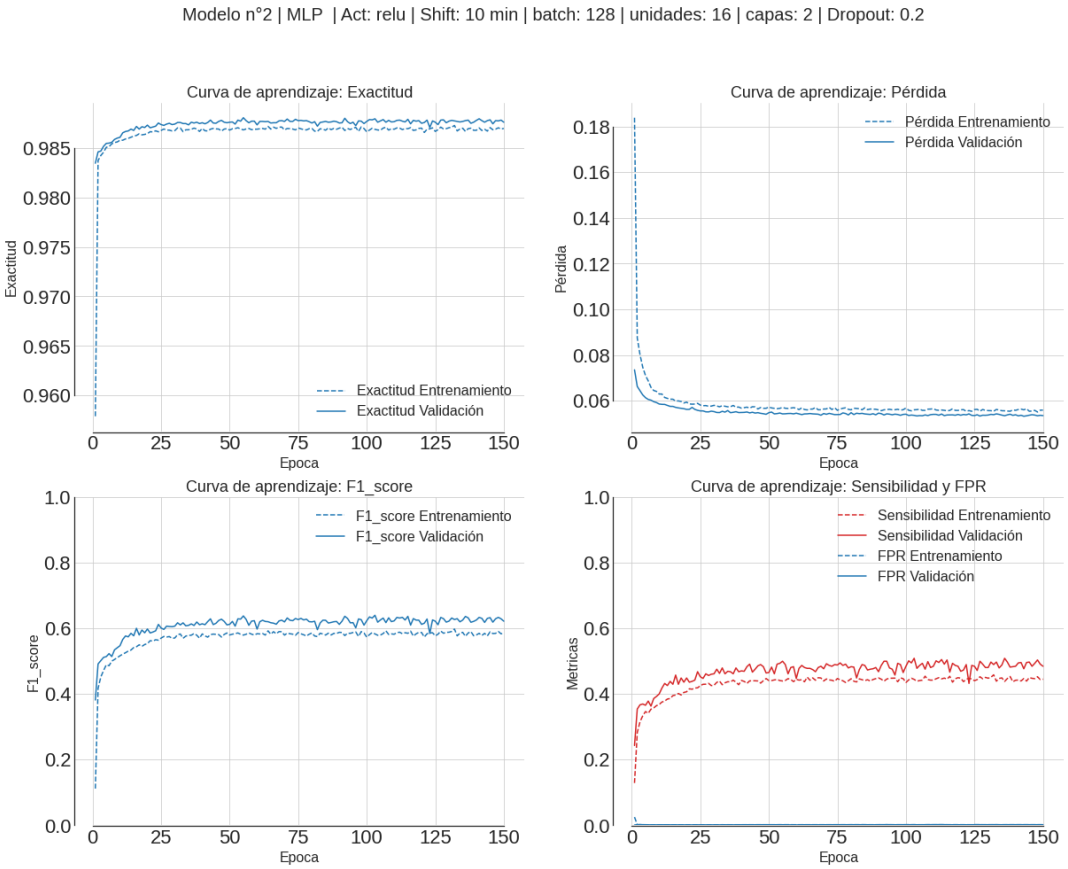


Figura 4.40: Curvas de aprendizaje caso II MLP.

Para reducir el sobreajuste de los modelos, una de las formas es añadir regularización por *dropout*. Los Casos II y III son exactamente iguales al Caso I pero con *dropout* a la salida de cada capa oculta.

Con un *dropout* de 0.2 (Figura 4.41) las curvas de pérdida ya no muestran el comportamiento observado en el Caso I, la pérdida de validación es casi idéntica a la de entrenamiento, por tanto, la incorporación de *dropout* habría eliminado el leve sobreajuste percibido en el Caso I. En general, las métricas del Caso II exhiben un peor rendimiento al obtenido en el Caso I.

Caso III

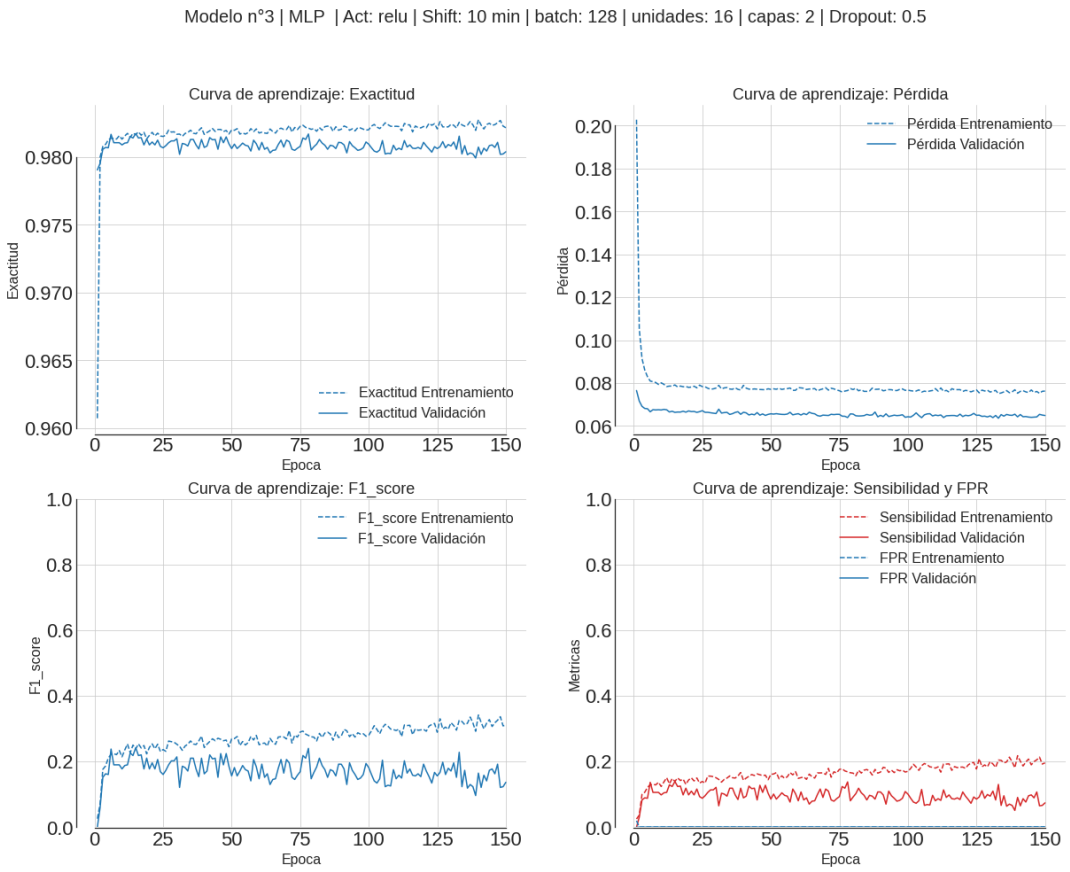


Figura 4.41: Curvas de aprendizaje caso III MLP.

La Figura 4.41 muestra los resultados al incorporar un d*ropout* de 0.5. El rendimiento del modelo disminuye dramáticamente como se puede observar sobre todo en las curvas de aprendizaje de F1-score y Sensibilidad. También se observa en todas las curvas que el modelo

está subajustado ya que el rendimiento en el conjunto de validación es claramente superior al del conjunto de entrenamiento, lo cual es consistente con la mayor influencia del *dropout*.

Caso IV

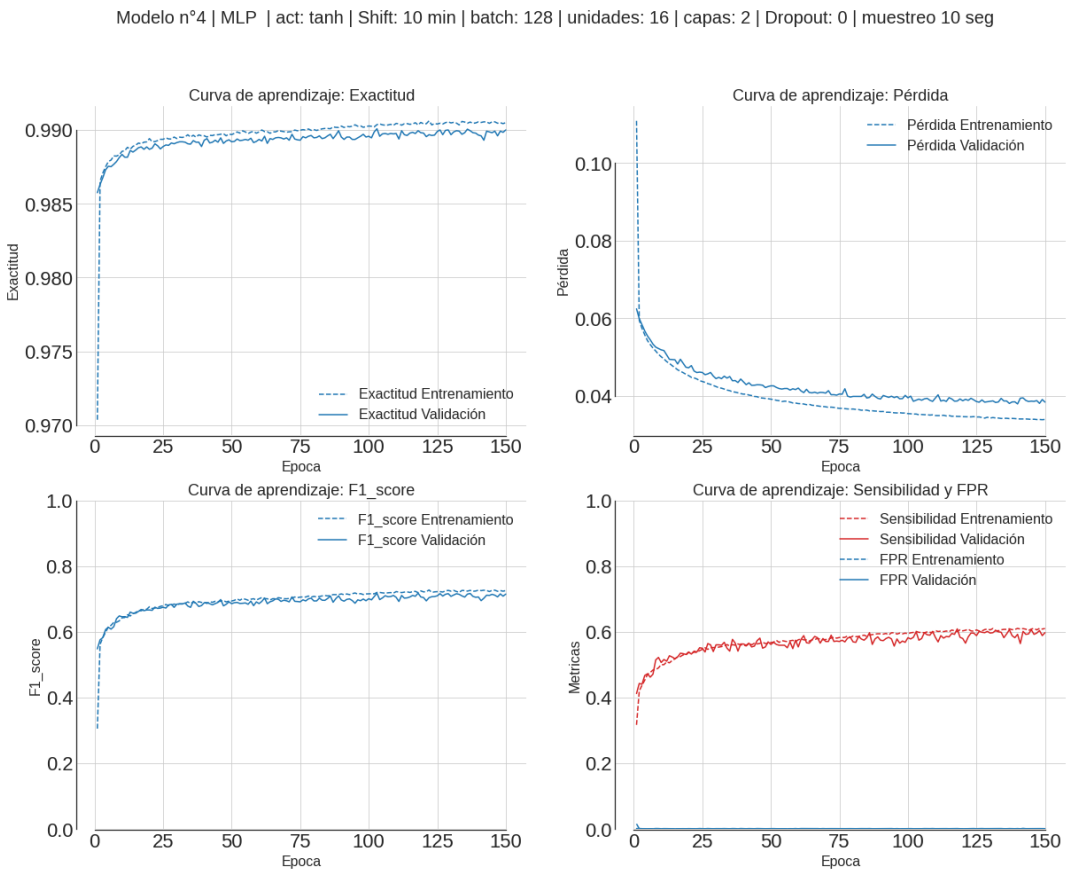


Figura 4.42: Curvas de aprendizaje caso IV MLP.

Este caso es análogo al Caso I pero con activación tangente hiperbólica. Los resultados (Figura 4.42) muestran curvas de aprendizaje prácticamente idénticas, indicando que la función de activación no tendría mayor influencia en el ajuste de los modelos. (al menos en este caso particular.

Caso V

En este caso se añade un método para combatir los efectos del desbalance de clases, donde se modifica la función de pérdida para penalizar en mayor medida cuando el modelo

no clasifica bien las muestras positivas (Clase 1) en comparación con las muestras negativas (Clase 0), como lo muestra la ecuación 4.3.

$$L(x) = -\frac{1}{n}\sum_{i=1}^{n}(w_1 y_i log(p_i) + w_0(1-y_i)log(1-p_i)) \quad (4.3)$$

Donde $w_1 > w_0$ son los pesos asignados a las muestras positivas y negativas respectivamente.

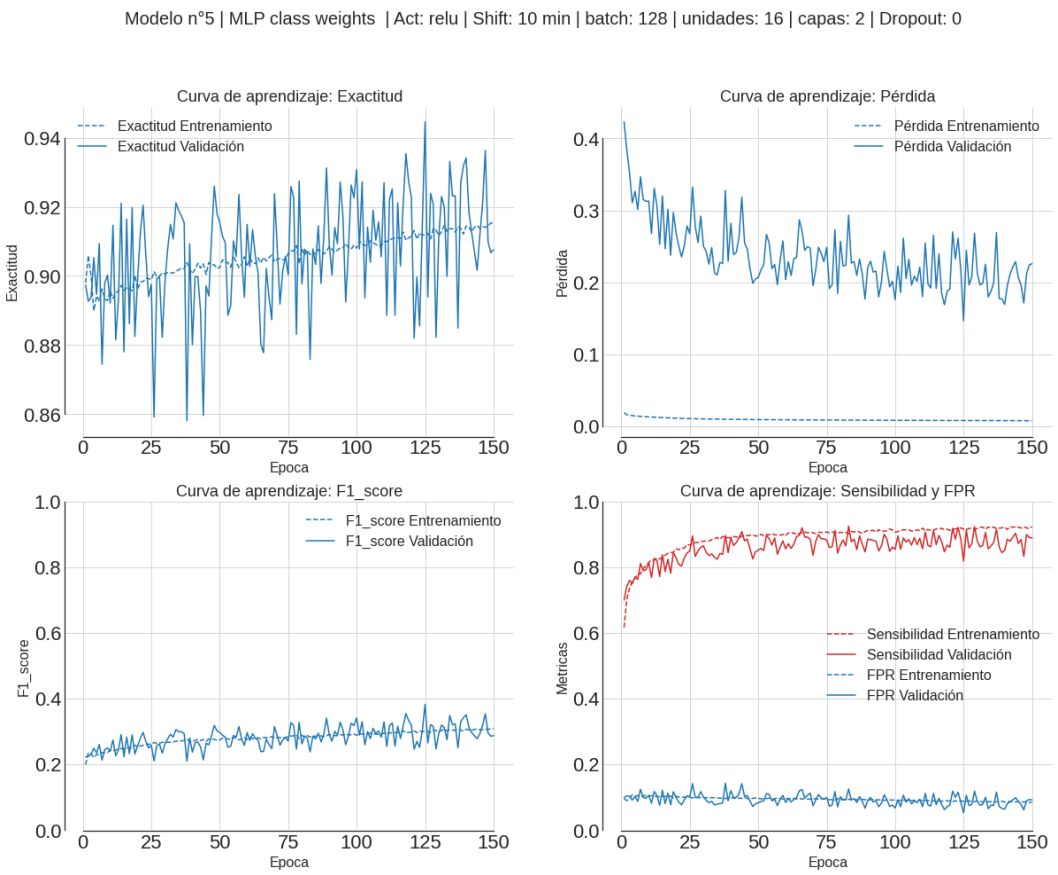


Figura 4.43: Curvas de aprendizaje caso V MLP

En primer lugar, las curvas de exactitud, en particular la de validación, muestra un comportamiento oscilatorio en torno a valores de 86% y 94% entre una época y otra. Esta oscilación también se observa en el resto de las métricas, en particular las calculadas con los datos de validación, lo cual podría indicar problemas en la convergencia del optimizador.

La curva de pérdida en el conjunto de validación está bastante alejada de la de entrenamiento, indicando sobreajuste.

Por otro lado, se obtiene una sensibilidad bastante alta (cercana a 0.9) indicando que el modelo es capaz de detectar la mayoría de las muestras positivas, sin embargo, el bajo rendimiento mostrado por la curva F1-score indica que la precisión es bastante pobre, es decir, el modelo está etiquetando las muestras como positivas sin mucha discriminación lo cual se traduce en un alto número de falsos positivos. Esto último es consistente con el aumento en la tasa de falsos positivos que llega a valores cercanos a 0.1.

Caso VI

Este caso incorpora regularización por *dropout* buscando reducir el sobreajuste del Caso V percibido en la pérdida del conjunto de validación.

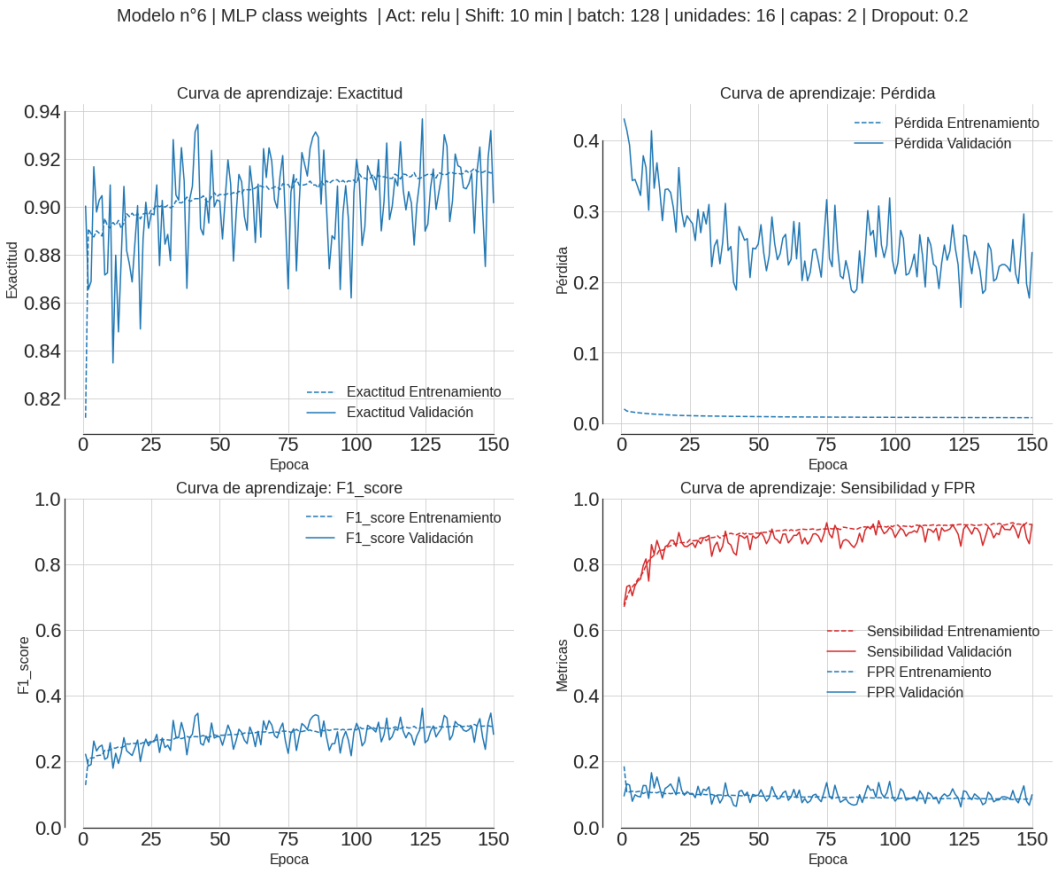


Figura 4.44: Curvas de aprendizaje caso VI MLP.

La Figura 4.44 muestra que la regularización por *dropout* no ayuda a reducir el sobreajuste en este caso, en efecto las curvas de aprendizaje son idénticas al Caso V.

Caso VII

Otra forma de combatir el desequilibrio de clases es incrementar la cantidad de muestras positivas (clase minoritaria). Una de las técnicas utilizadas consiste en sintetizar muestras positivas, para ello se ocupa un método llamado *Synthetic Minority Over-Sampling Technique*, o SMOTE por sus siglas en inglés.

Básicamente las nuevas muestras son creadas usando interpolación lineal entre muestras de la clase minoritaria.

Se crean muestras hasta que las clases quedan perfectamente balanceadas. Cabe destacar que este método solo se aplica a las muestras del conjunto de entrenamiento.

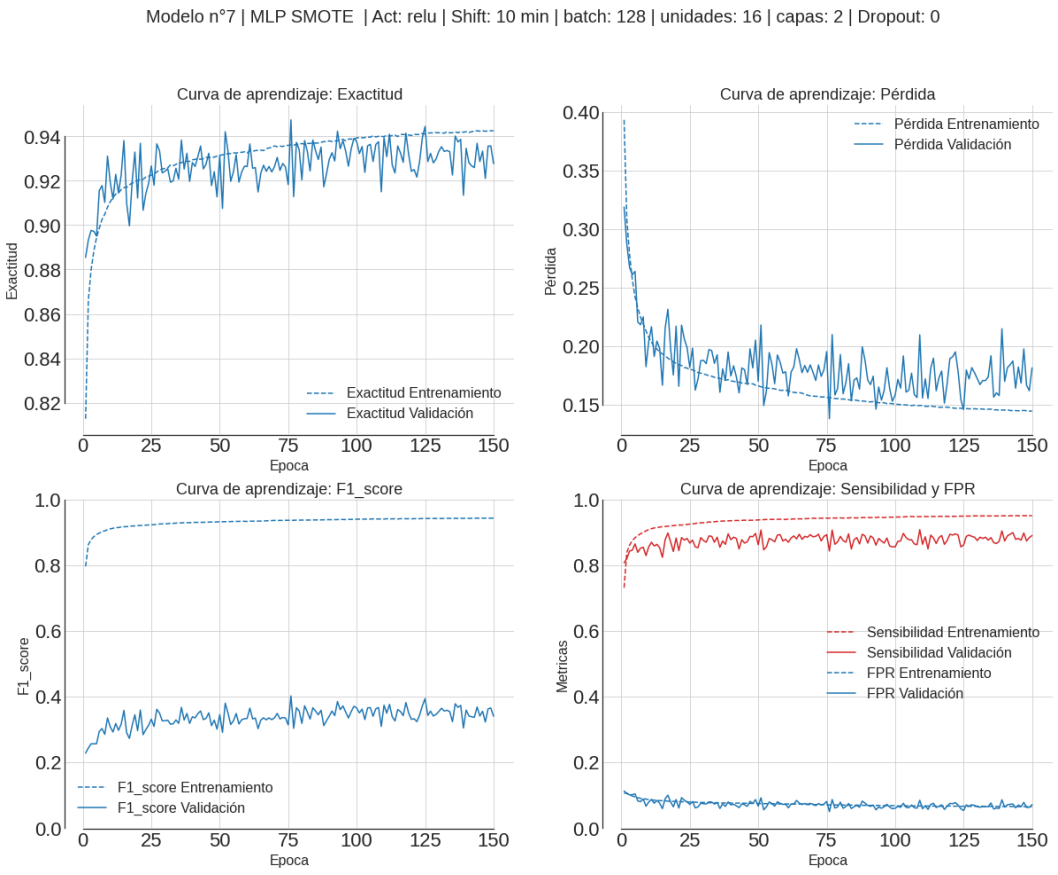


Figura 4.45: Curvas de aprendizaje caso VII MLP

Similar al Caso anterior, se observa el carácter oscilatorio en las curvas de aprendizaje relativas al conjunto de validación. Otro aspecto similar es la alta sensibilidad (cercana a 0.9) que indica que el modelo etiqueta correctamente el 90% de las muestras positivas, sin embargo, la cantidad de falsos positivos sigue siendo alta, como lo indica la tasa de falsos positivos (FPR).

A diferencia de los casos V y VI, la curva de aprendizaje del F1-score asociada al conjunto de validación sigue en valores cercanos a 0.4, mientras que la curva asociada al conjunto de entrenamiento indicaría un rendimiento altísimo (>90%) en esta métrica. Esta particularidad podría deberse a sobrajuste, pero no se puede asegurar ya que las otras curvas de aprendizaje no exhiben este notorio comportamiento.

Caso VIII

Se aplicó regularización por *dropout* para comprobar si efectivamente existía sobreajuste y si de esta manera era posible obtener un F1-score mayor con los datos de validación.

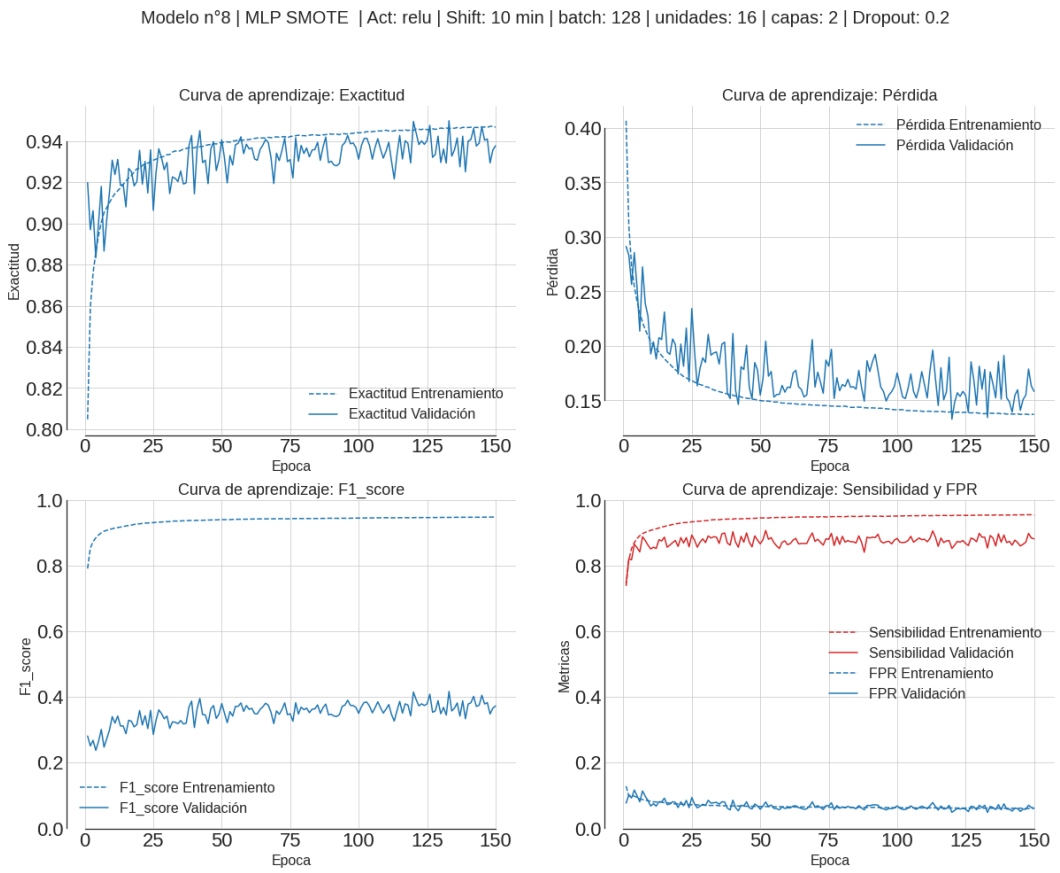


Figura 4.46: Curvas de aprendizaje caso VIII MLP.

Como muestra la Figura 4.46, los resultados no cambian y las curvas de aprendizaje son idénticas al caso sin *dropout*.

Caso IX

El Caso IX es similar a los casos I y IV. La función de activación utilizada aquí es la *Scaled Exponential Linear Unit* (*selu*), con un AlphaDropout de 0.1.

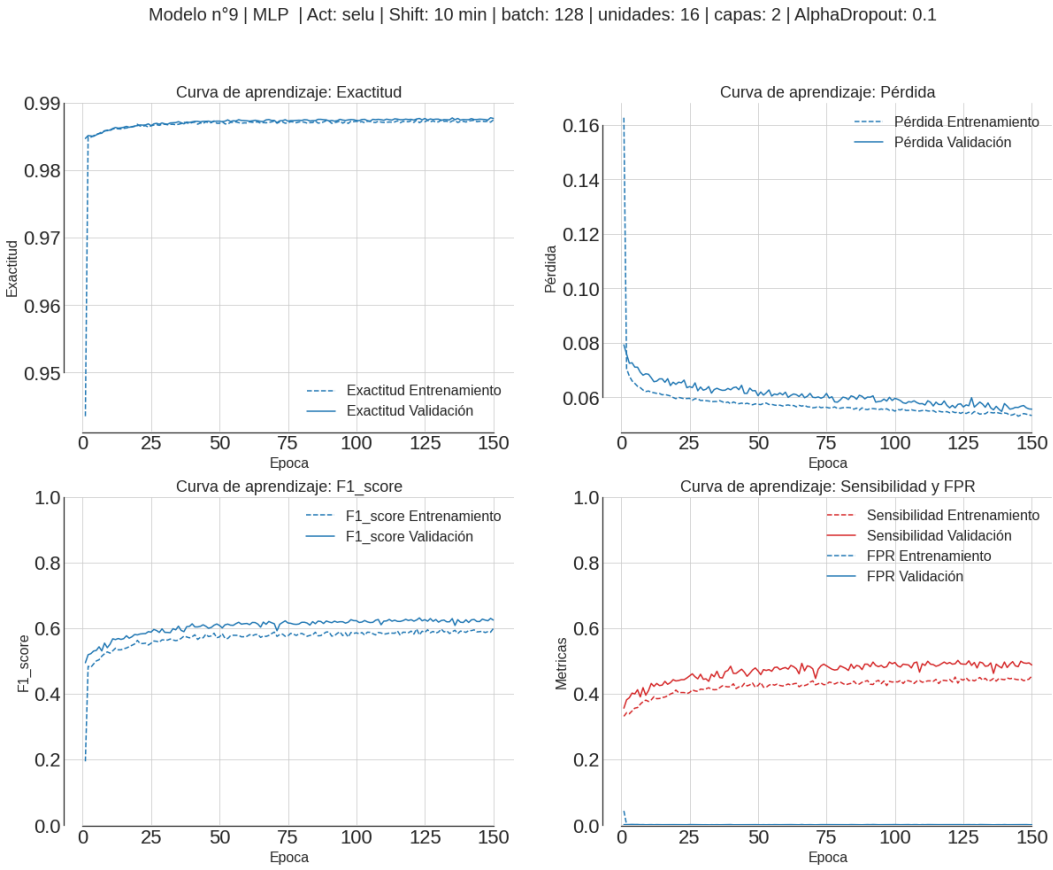


Figura 4.47: Curvas de aprendizaje caso IX MLP.

Los resultados son relativamente similares a los obtenidos con funciones de activación *relu* y *tanh*, aunque el rendimiento mostrado por algunas métricas, como la sensibilidad y F1-score, es peor en comparación a los casos anteriores.

Evaluación en el conjunto de testeo

Finalmente, los 9 casos son evaluados en el conjunto de testeo. En la Tabla 4.21 se muestran los resultados de esta evaluación.

Tabla 4.21: Resultados de evaluación de modelos MLP en conjunto de testeo.

Caso	Exactitud	Pérdida	F1-score	Sensibilidad	FPR
I	0.9891	0.0388	0.6868	0.5563	0.0014
II	0.9879	0.0534	0.6367	0.4935	0.0012
III	0.9799	0.0663	0.1351	0.0728	0.0001
IV	0.9897	0.0372	0.7121	0.5942	0.0016
V	0.9068	0.2269	0.2946	0.9063	0.0932
VI	0.9019	0.2362	0.2861	0.9143	0.0983
VII	0.9266	0.1805	0.3511	0.9242	0.0733
VIII	0.9384	0.1592	0.3880	0.9093	0.0610
IX	0.9879	0.0561	0.6357	0.4905	0.0012

De acuerdo con los resultados, los métodos de balance de clases (Casos V, VI, VII y VIII) aumentan el rendimiento de los modelos en términos de la métrica sensibilidad, pasando del 50% a sobre el 90% como muestra la Tabla 4.21. Sin embargo, este buen rendimiento se obtiene a costa de una alta presencia de falsos positivos como evidencian las métricas FPR, con valores cercanos a 0.1, y el F1-score con valores menores a 0.4.

La regularización por *dropout* no mostró grandes mejoras en los modelos entrenados, aunque se debe mencionar que ninguno de los casos mostraba un claro sobreajuste, de modo que no se puede concluir sobre la capacidad del *dropout* para reducir el sobreajuste. Por otro lado, valores muy altos de *dropout* (0.5) son contraproducentes y empeoran el rendimiento de los modelos. Si se hace uso de regularización por d*ropout* es mejor utilizar valores moderados (0.2)

La función de activación no parece ser un factor que influye demasiado en el rendimiento de los modelos.

En general las curvas de rendimiento no indican sobreajuste de los modelos para la mayoría de los casos, lo que da paso a desarrollar modelos más complejos con la finalidad de mejorar su rendimiento.

Iteración II

En virtud de lo anterior, se realizó una segunda iteración donde adicionalmente se ajustaron otros parámetros no se habían ajustado en la primera iteración.

Los parámetros que se ajustaron se muestran a continuación donde también se especifica qué valores tomó cada uno.

- Cantidad de capas ocultas: [2,3,4]
- Cantidad de neuronas: [64,32,16,8]
- Muestras positivas (minutos previos al evento): [10, 20, 30, 120, 180]
- Activación: [relu, tanh]
- Dropout: [0.0, 0.2, 0.3, 0.5]

Debido a las innumerables combinaciones de parámetros, se evaluaron combinaciones aleatorias de dichos parámetros.

De esta manera se entrenaron un total de 38 modelos, de los cuales 20 fueron seleccionados por sus buenos resultados.

La selección de los mejores modelos fue hecha en base a las curvas de aprendizaje, en particular las asociadas a la pérdida, F1-score, sensibilidad y FPR.

Se tomaron en consideración aspectos como el sobreajuste y subajuste de los modelos, comparando las curvas de aprendizaje del conjunto de entrenamiento con las curvas del conjunto de validación. También se consideró la convergencia de las curvas hacia un determinado valor y el comportamiento decreciente de la curva de pérdida entre otros aspectos.

Los 20 modelos seleccionados fueron evaluados en el conjunto de testeo, de los cuales 3 destacaron en distintas métricas.

Caso I: Modelo con máxima sensibilidad.

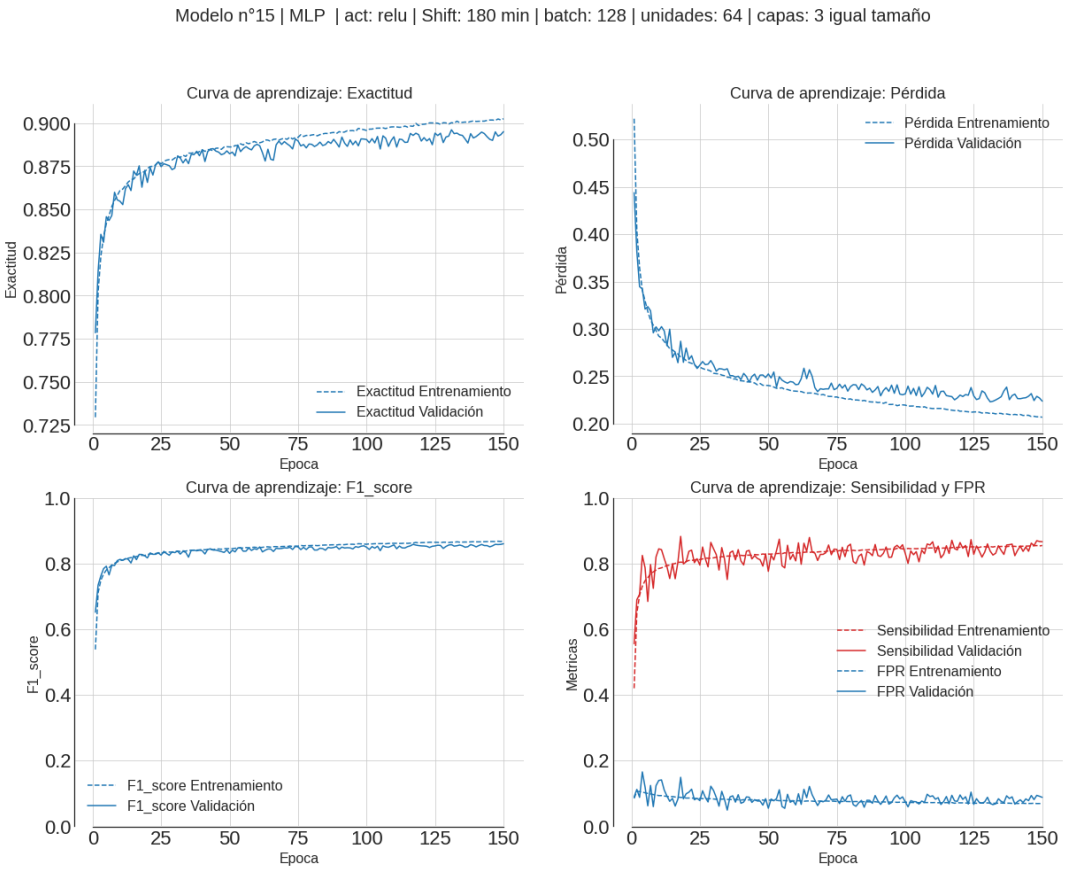


Figura 4.48: Modelo Iteración II con máxima sensibilidad.

Tabla 4.22: Métricas modelo Iteración II con máxima sensibilidad en conjunto testeo.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1.9961	0.6400	0.9761	0.1044	0.3674

Las curvas de aprendizaje obtenidas a partir de los datos de entrenamiento y validación indican un buen rendimiento del modelo, al evaluar en el conjunto de testeo, se obtiene la máxima sensibilidad (~98%) pero el F1-score es de apenas 0.1 y el FPR cercano a 0.4, indicando que existe una enorme cantidad de falsos positivos. La baja Exactitud (64%) también indica que el modelo tiene problemas etiquetando correctamente la clase negativa (Clase 0), por tanto, la alta sensibilidad obtenida es debido a que el modelo está clasificando la mayoría de las muestras como positivas (Clase 1).

El alto número de falsos positivos puede verse claramente al graficar señales temporales evaluadas por el modelo como las mostradas en la Figura 4.49. Estas señales corresponden a 5 eventos del conjunto de los 81 eventos estables y aislados. En color rojo las muestras clasificadas positivas, en azul las negativas, y la franja roja vertical indica los minutos previos al evento que se consideran como precursor, en otras palabras, las muestras positivas reales.

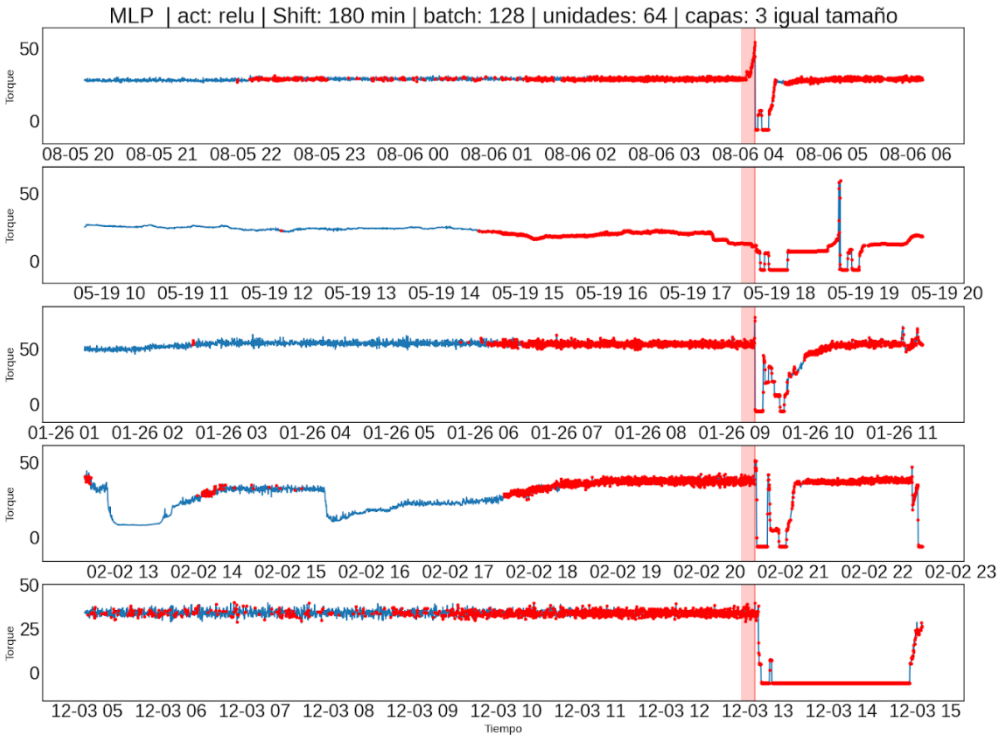


Figura 4.49: Señales temporales de algunos eventos evaluadas en modelo Iteración II con máxima sensibilidad.

Caso II: Modelo con pérdida mínima, exactitud y F1-score máximos.

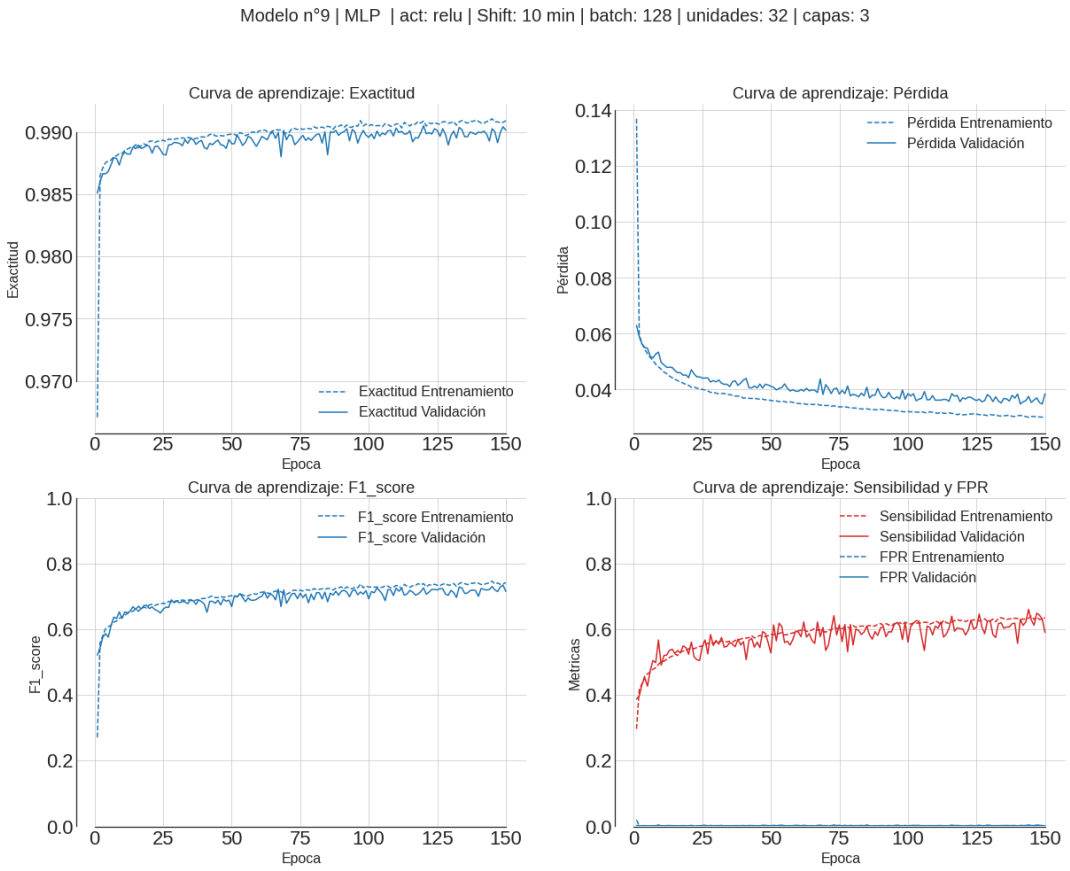


Figura 4.50: Modelo iteración II con pérdida mínima, exactitud y F1-scores máximos.

Tabla 4.23: Métricas modelo Iteración II con pérdida mínima, exactitud y F1-scores máximos en conjunto testeo.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.0386	0.9894	0.5813	0.7028	0.0016

Este modelo resulta ser bastante más equilibrado. Como se puede observar en la Figura 4.50 y la Tabla 4.23, las métricas calculadas tanto en conjuntos de entrenamiento, validación y testeo tienen valores similares.

El rendimiento en señales temporales se puede ver en la Figura 4.51, donde se observa que el modelo reconoce los peaks de torque, pero en varios casos no se anticipa lo suficiente.

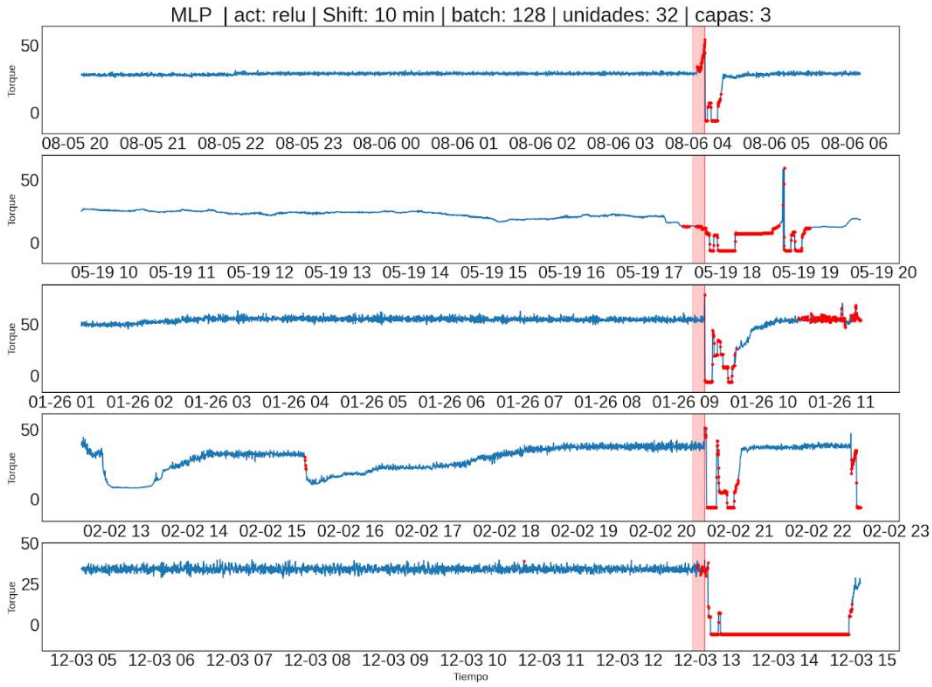


Figura 4.51: Señales temporales de algunos eventos evaluadas en modelo iteración II con pérdida mínima, exactitud y F1-scores máximos.

Caso III: Modelo con FPR mínimo.

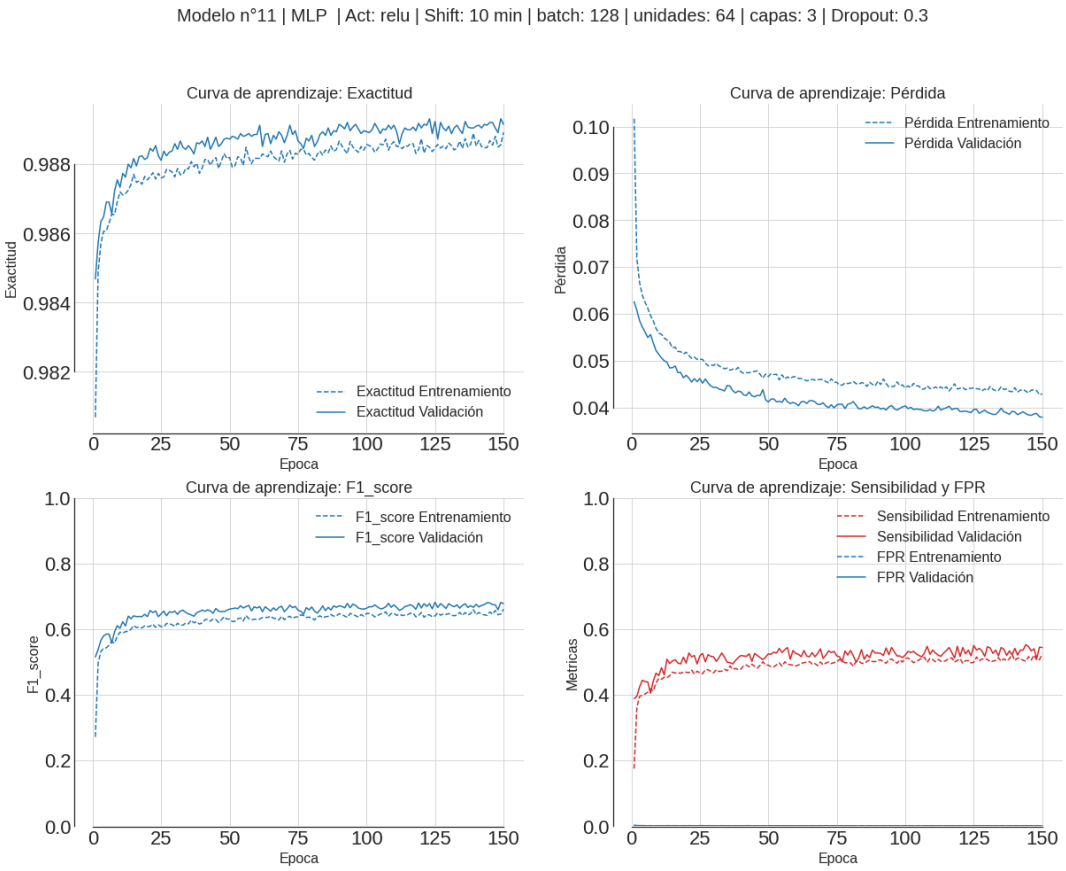


Figura 4.52: Modelo iteración II con FPR mínimo.

Tabla 4.24: Métricas modelo Iteración II con FPR mínimo en conjunto testeo.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.0377	0.9895	0.5653	0.6991	0.0011

Este modelo al igual que el caso II es equilibrado, con métricas calculadas en conjuntos de entrenamiento, validación y testeo bastante idénticas, pero sensiblemente inferiores al caso II, de manera que el mejor modelo de la Iteración II es el caso II.

Iteración III

Una última iteración consiste en tomar la arquitectura del mejor modelo de la iteración anterior y evaluar qué tanto afecta aumentar las muestras positivas previas al evento y evaluar así la capacidad del modelo para anticipar un evento de alto torque. Para ello se evaluaron los siguientes casos.

- Muestras positivas (minutos previos al evento): [10, 30, 60, 180]

Con muestras positivas de 10, 30 y 60 minutos previos al evento, se obtienen curvas de aprendizaje bastante similares (ver Anexo), con métricas de sensibilidad cercanas a 0.6 y F1-score en torno a 0.7.

Las curvas de aprendizaje para muestras positivas de 180 minutos previos son notablemente superiores a los 3 casos anteriores, con valores de 0.8 de sensibilidad y F1-score. Sin embargo, este buen desempeño no se ve reflejado en el conjunto de testeo (Tabla 4.25), ya que las métricas F1-score y FPR evidencian una alta presencia de falsos positivos, muy similar a lo ocurrido en el caso b) de la iteración II.

Tabla 4.25: Métricas Modelos iteración III en conjunto testeo.

Muestras positivas previas [min]	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
10	0.0319	0.9906	0.6321	0.7433	0.0015
30	0.1273	0.9726	0.7916	0.5541	0.0234
60	0.3209	0.9181	0.8913	0.3185	0.0814
180	1.4010	0.6474	0.9492	0.1037	0.3592

Al observar detenidamente la Tabla 4.25 podemos ver claramente cómo la Exactitud y el F1-score disminuyen con el aumento de muestras positivas reales. Por otro lado, la sensibilidad, pérdida y FPR aumentan con el aumento de muestras positivas reales.

En conclusión, al aumentar la cantidad de muestras etiquetadas como positivas reales antes de la ocurrencia del evento de alto torque, se obtiene un aumento en la sensibilidad,

que es un efecto deseado ya que el modelo en teoría estaría aparentemente mejorando su capacidad de predicción.

Sin embargo, este aumento en la sensibilidad viene de la mano con un aumento indeseado de falsos positivos, evidenciado por el aumento radical de la tasa de falsos positivos (FPR) y el rápido descenso de la métrica F1-score.

Modelo base definitivo

De acuerdo con los resultados anteriores, el modelo base más equilibrado se obtiene para muestras positivas 10 minutos antes del evento. A continuación, se muestra la arquitectura de este modelo y sus resultados

.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hidden_layer_1 (Dense)      (None, 32)                608

 hidden_layer_2 (Dense)      (None, 16)                528

 hidden_layer_3 (Dense)      (None, 8)                 136

 output_layer (Dense)        (None, 1)                 9

=================================================================
Total params: 1,281
Trainable params: 1,281
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.53: Arquitectura modelo definitivo MLP

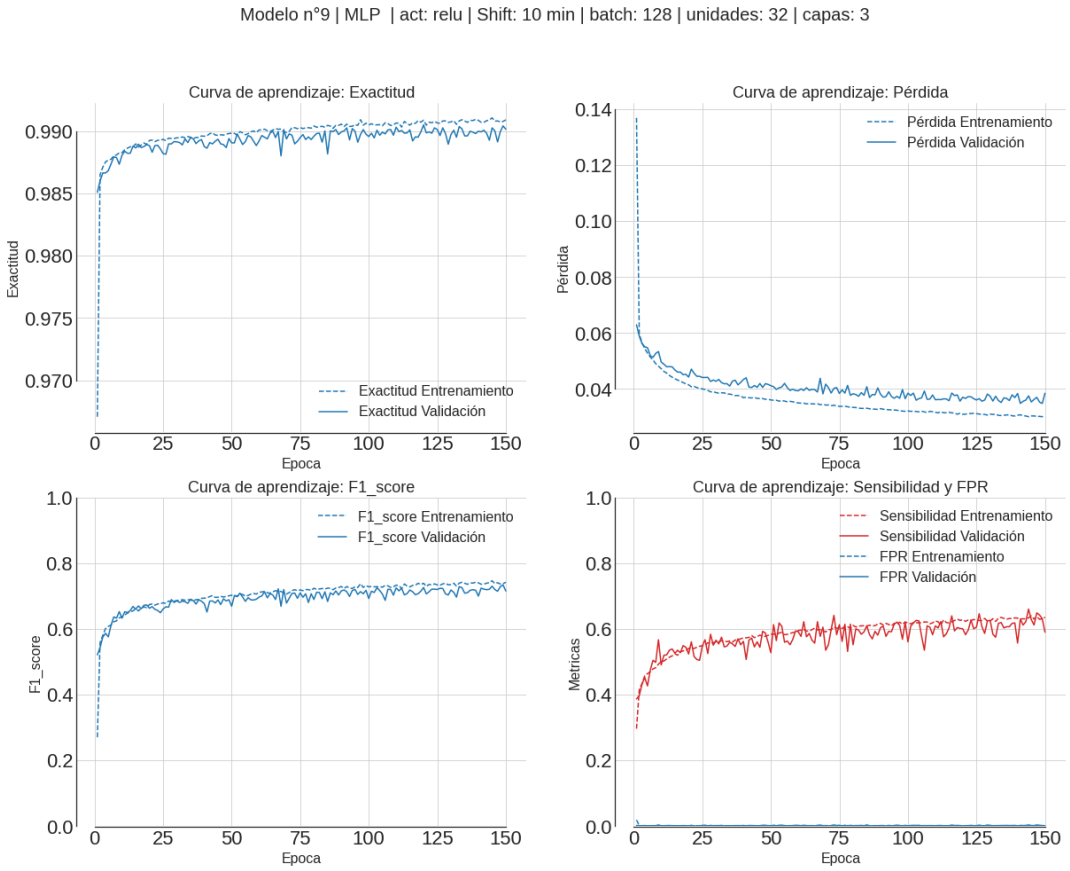


Figura 4.54: Curvas de aprendizaje modelo definitivo MLP.

Tabla 4.26: Métricas conjunto de testeo modelo definitivo MLP.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.0386	0.9894	0.5813	0.7028	0.0016

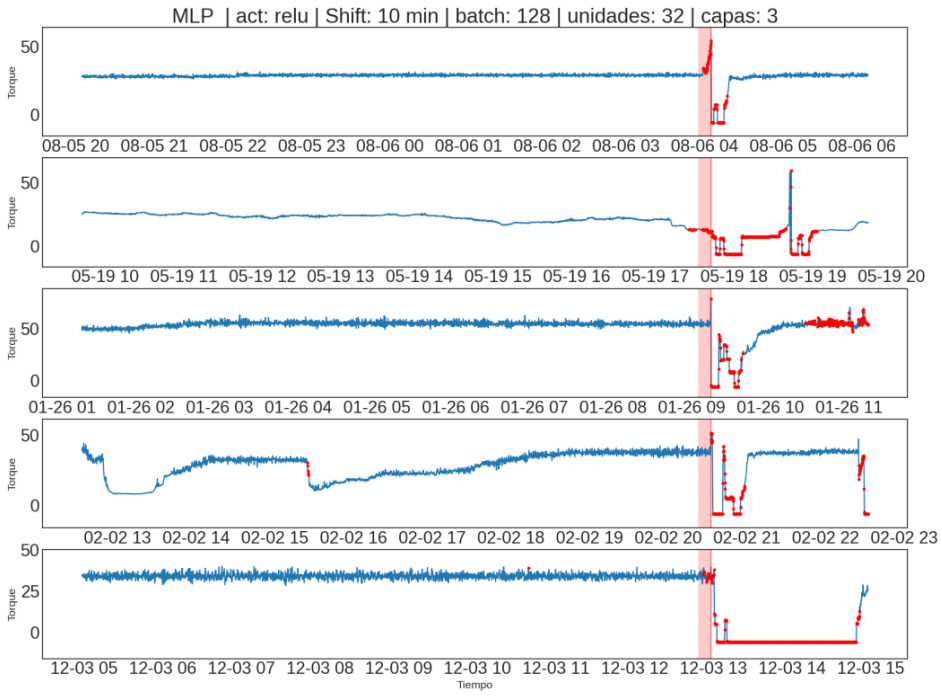


Figura 4.55: Señales temporales de algunos eventos evaluadas en modelo definitivo MLP.

La Figura 4.55 muestra algunas señales temporales que corresponde a 5 de los 81 eventos del conjunto de datos con que fueron entrenados los modelos. No se puede asegurar a qué conjunto de datos pertenece cada señal puesto que los datos fueron seleccionados de forma aleatoria para formar los conjuntos de entrenamiento, validación y testo, de modo que cada señal temporal puede estar compuesta de datos pertenecientes a los 3 conjuntos.

La Figura 4.56 muestra algunas señales temporales que contienen eventos que fueron descartados en un principio y por lo tanto no son parte del conjunto de datos con que fueron entrenados los modelos. Utilizando estos datos se puede ver la capacidad de generalización del modelo en datos totalmente nuevos.

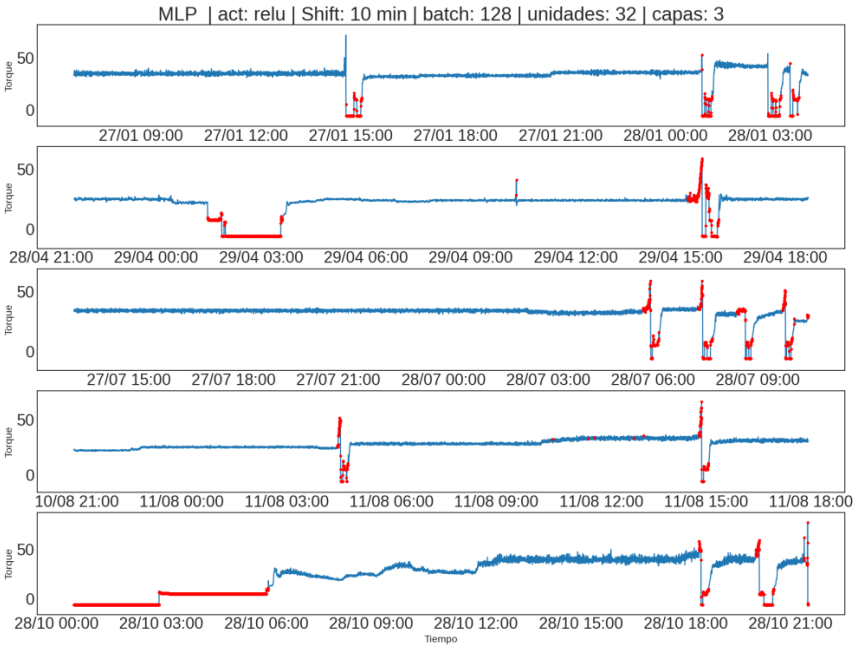


Figura 4.56: Señales temporales evaluadas en modelo definitivo MLP.

Recordemos que estos modelos no consideran relaciones temporales, las muestras son clasificadas por el modelo independientemente. Esto puede explicar el hecho de que los modelos *MLP* no son capaces de anticipar los eventos de alto torque, y solamente pueden reconocer los peaks o cualquier comportamiento extraño instantáneo en las señales, como muestran las Figuras 4.55 y 4.56.

4.3.7. Modelos con capas Recurrentes y Convolucionales: Versión I

Los modelos tanto con capas Recurrentes como Convolucionales se manejan de igual forma en el procesamiento de datos. En una primera etapa los datos fueron procesados siguiendo la siguiente metodología.

1. Reestructuración de datos en una única señal.
2. Etiquetado de los datos.
3. Disminuir frecuencia de muestreo.
4. Temporalizar datos.
5. Dividir datos temporalizados aleatoriamente en conjuntos de Entrenamiento, Validación y Testeo.
6. Normalizar datos a partir de datos de Entrenamiento.

Modelos con capas Recurrentes

Iteración I

Como primera iteración se prueban diferentes variantes de modelos con capas *LSTM* con una arquitectura similar a la de la Figura 4.57.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_layer_1 (LSTM)         (None, 30, 16)            2240

 dropout (Dropout)           (None, 30, 16)            0

 lstm_layer_2 (LSTM)         (None, 30, 8)             800

 flatten (Flatten)           (None, 240)               0

 dropout_1 (Dropout)         (None, 240)               0

 output (Dense)              (None, 1)                 241

=================================================================
Total params: 3,281
Trainable params: 3,281
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.57: Arquitectura modelos LSTM de la Iteración I.

Como muestra la Figura 4.57, las características de los modelos son:

- Capa de entrada: Tiene tantas unidades como parámetros (18)
- Capas ocultas: 2 capas LSTM de 16 y 8 unidades respectivamente.
- Capa de salida: Una sola unidad con activación Sigmoide.

Para el entrenamiento se utilizan los siguientes parámetros:

- Tamaño de batch: 128
- Épocas: 100
- Optimizador: Adam
- Learning rate: 0.001
- Función de pérdida: Binary Cross Entropy
- Muestras positivas: 10 minutos
- Tamaño ventana temporalización: 30 minutos
- Función de Activación Recurrente: Sigmoide

Tabla 4.27: Modelos LSTM entrenados en la Iteración I.

Caso	Activación	Dropout	Dropout recurrente	Estructura
I	tanh	0.0	0.0	Restricted Stateless
II	relu	0.0	0.0	Restricted Stateless
III	tanh	0.0	0.0	Unrestricted Stateless
IV	tanh	0.5	0.5	Unrestricted Stateless
V	tanh	0.0	0.0	Unrestricted Stateless Backwards
VI	tanh	0.5	0.0	Unrestricted Stateless Bidirectional
VII	tanh	0.0	0.0	Unrestricted Statefull

Caso I

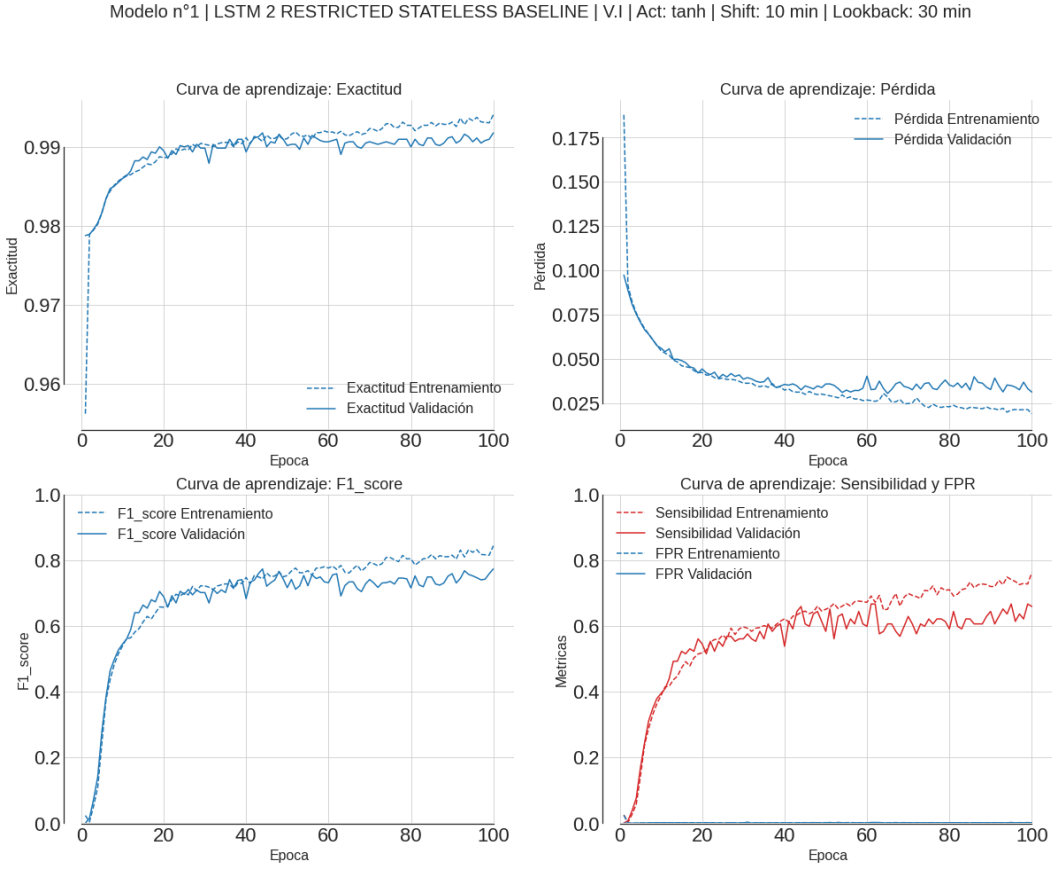


Figura 4.58: Curvas de aprendizaje caso I modelos LSTM Iteración I.

Las curvas de aprendizaje de la Figura 4.58 muestran una mejora respecto a la mayoría de los modelos *MLP* desarrollados anteriormente. El comportamiento de la curva de pérdida indica que el modelo converge, solo con leves indicios de sobreajuste que pueden ser apreciados en el resto de las curvas.

Caso II

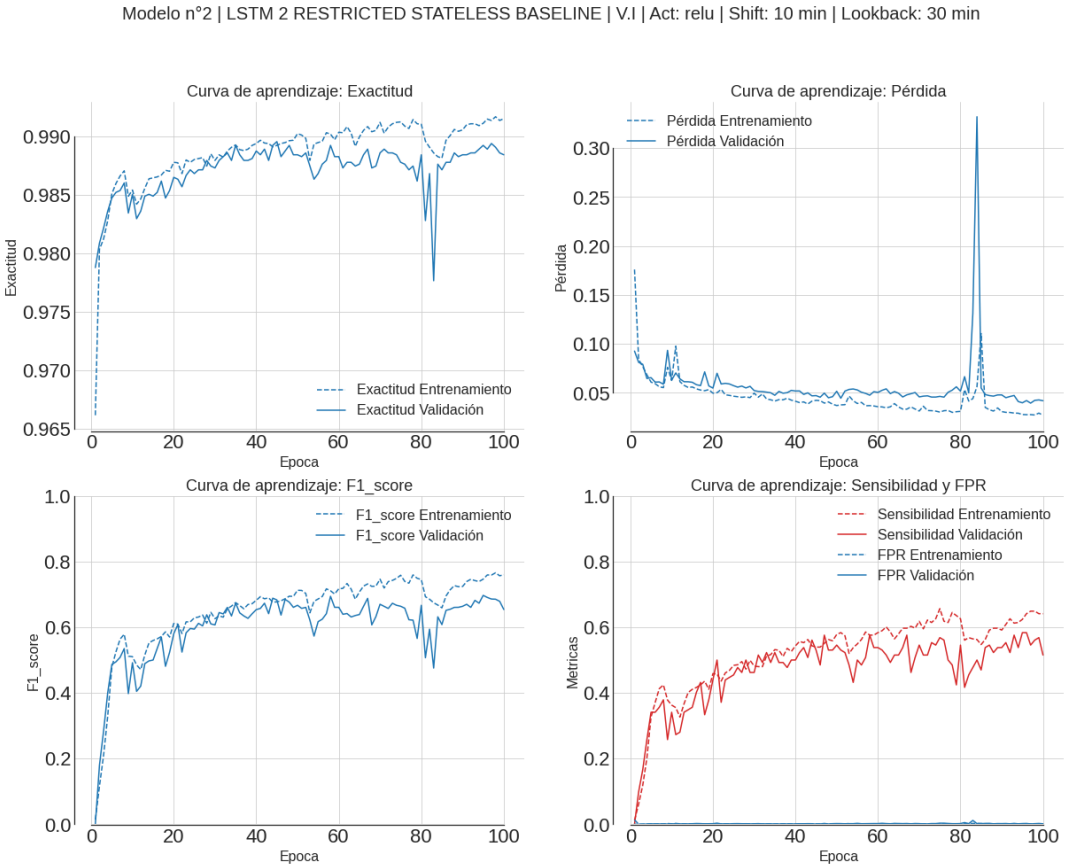


Figura 4.59: Curvas de aprendizaje caso II modelos LSTM Iteración I.

Este caso es exactamente igual al anterior, pero con activación *relu*. La Figura 4.59 muestra resultados sensiblemente inferiores a los obtenidos con una activación *tanh.* Esta última es usada preferentemente en los modelos posteriores.

Caso III

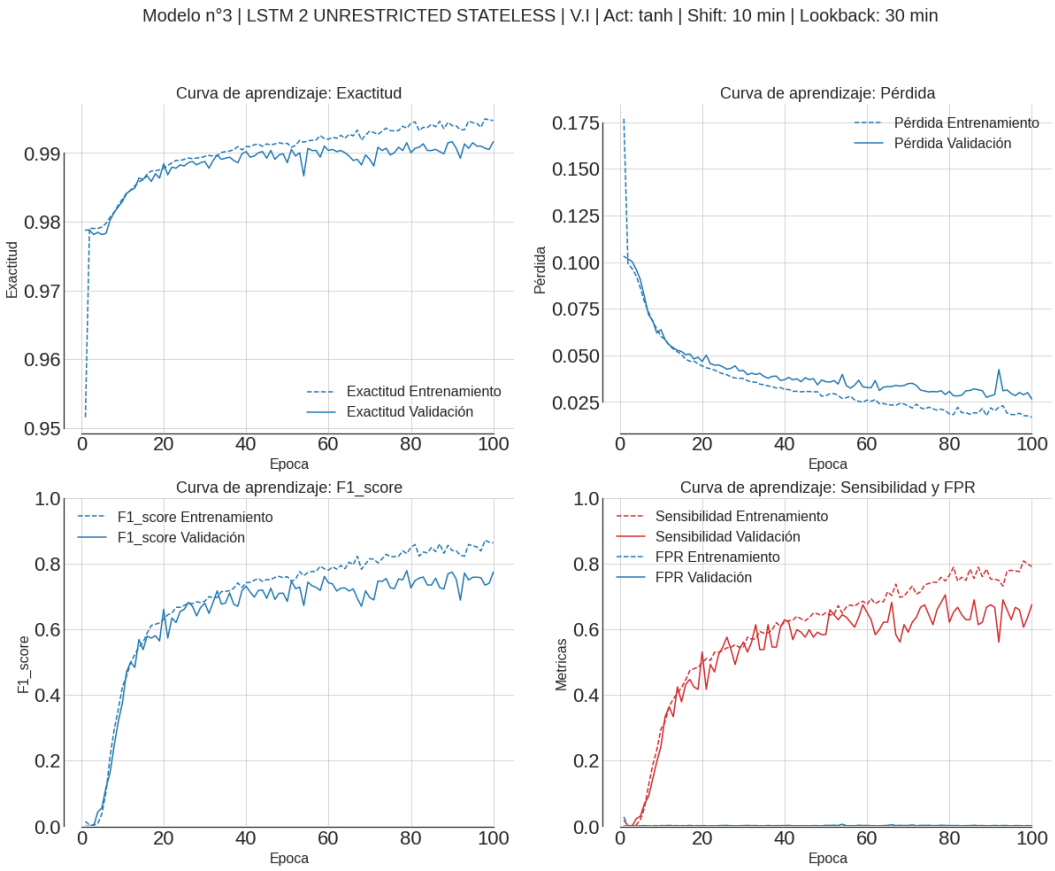


Figura 4.60: Curvas de aprendizaje caso III modelos LSTM Iteración I.

Los resultados de una red *LSTM unrestricted* mostrados en la Figura 4.60 parecen ser idénticos a su contraparte *restricted* (Figura 4.58), al menos en términos de sus curvas de aprendizaje, lo cuál indicaría que esta variante no parece influir en el desempeño de los modelos para esta aplicación en particular.

Caso IV

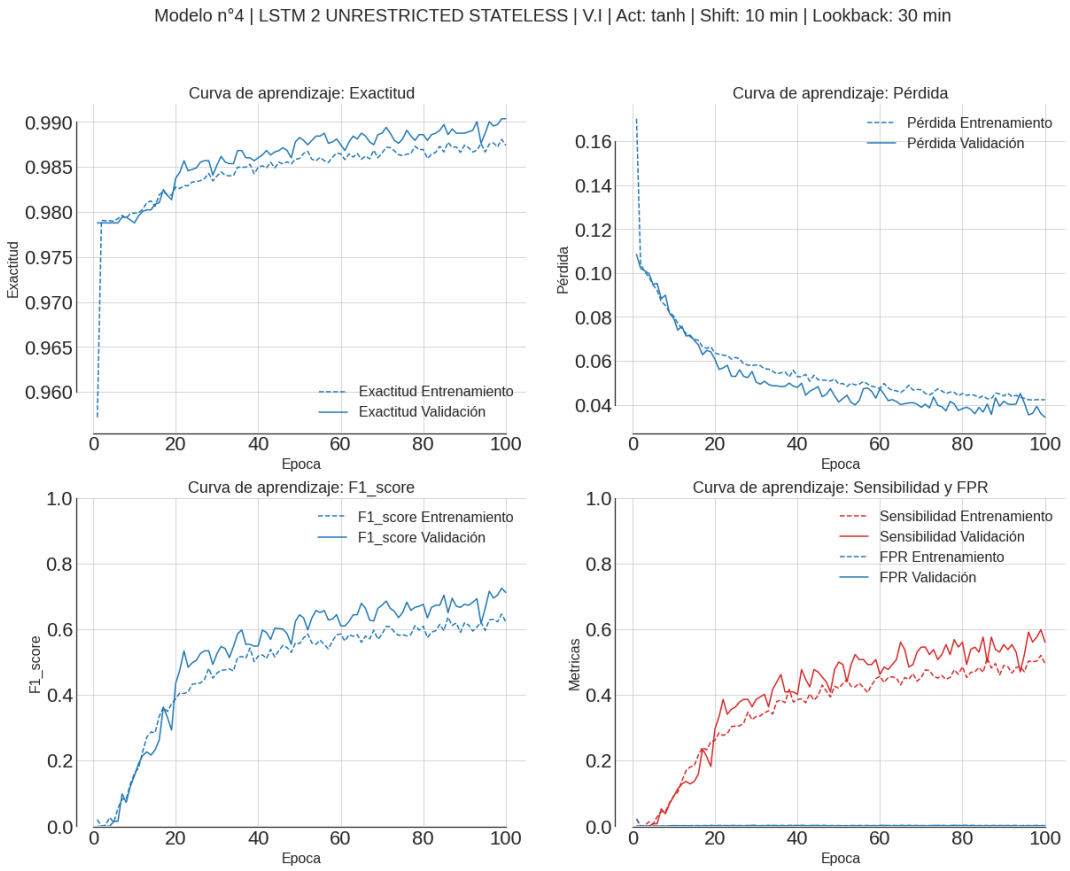


Figura 4.61: Curvas de aprendizaje caso IV modelos LSTM Iteración I.

Los casos I y III mostraban indicios de sobreajuste, por este motivo se añade regularización por *dropout* (0.5). Junto con esto se agrega *recurrent dropout* (0.5), que consiste en desactivar aleatoriamente una fracción de los *inputs* que ingresan a los estados recurrentes de las células *LSTM.*

Las curvas de la Figura 4.61 ya no muestran los indicios de sobreajuste que se percibieron en los casos anteriores, incluso ahora se observa un leve subajuste producto del alto valor de *dropout* utilizado.

Caso V

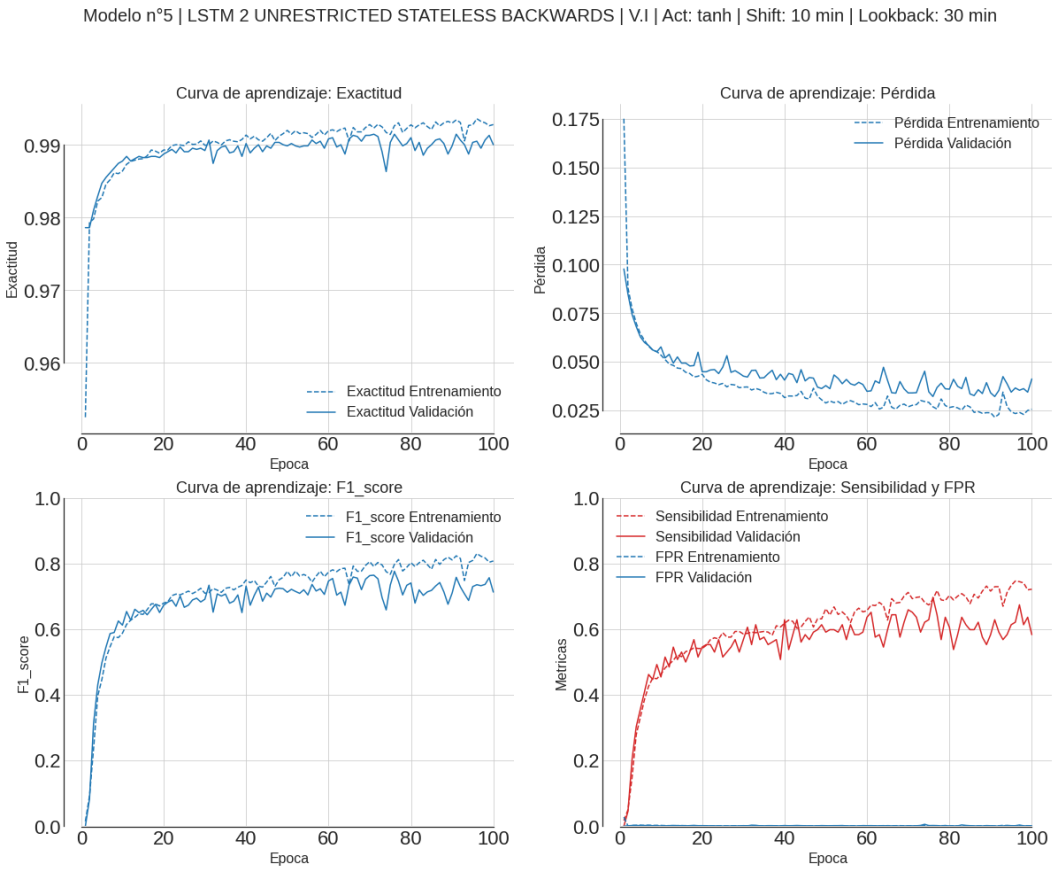


Figura 4.62: Curvas de aprendizaje caso V modelos LSTM Iteración I.

La variante *LSTM backwards* tampoco parece aportar al rendimiento del modelo, como muestra la Figura 4.62 las curvas de aprendizaje son idénticas a los casos I y III.

Caso VI

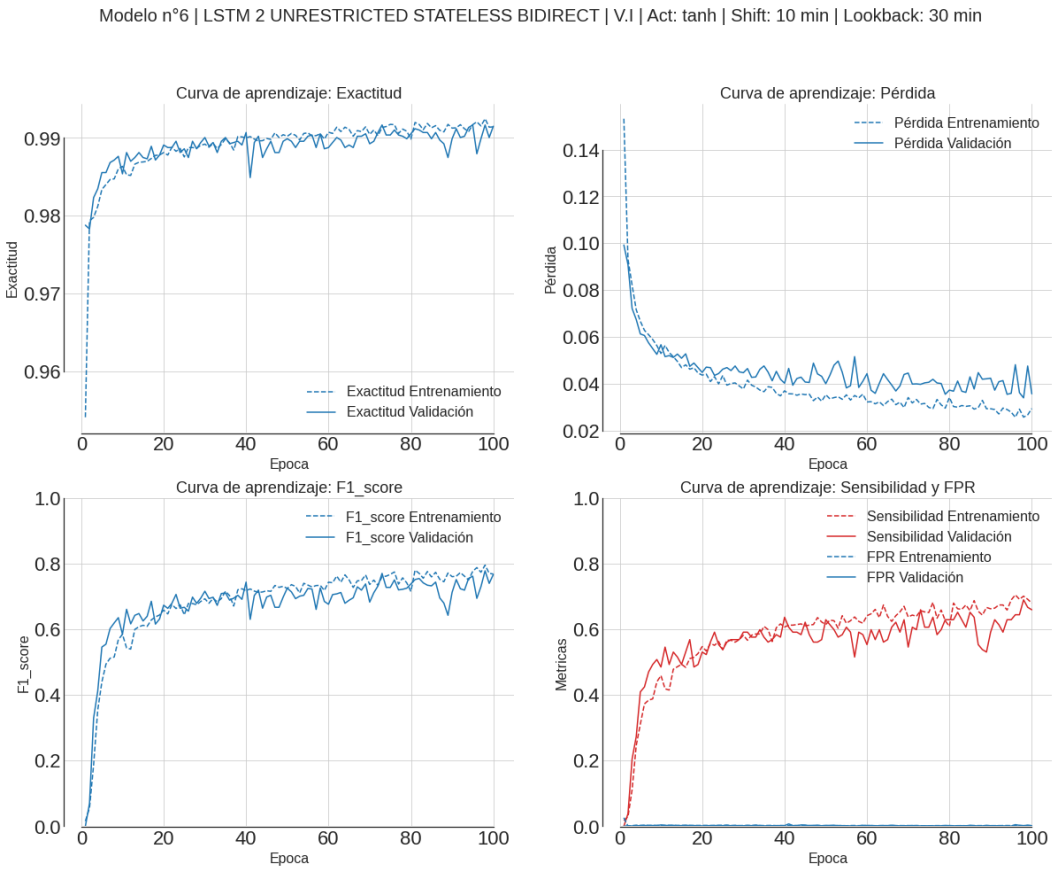


Figura 4.63 Curvas de aprendizaje caso VI modelos LSTM Iteración I.

La variante *LSTM Bidirectional* también exhibe un rendimiento similar a los casos previos, aunque las curvas de aprendizaje (Figura 4.63) no muestran indicios de sobreajuste.

Caso VII

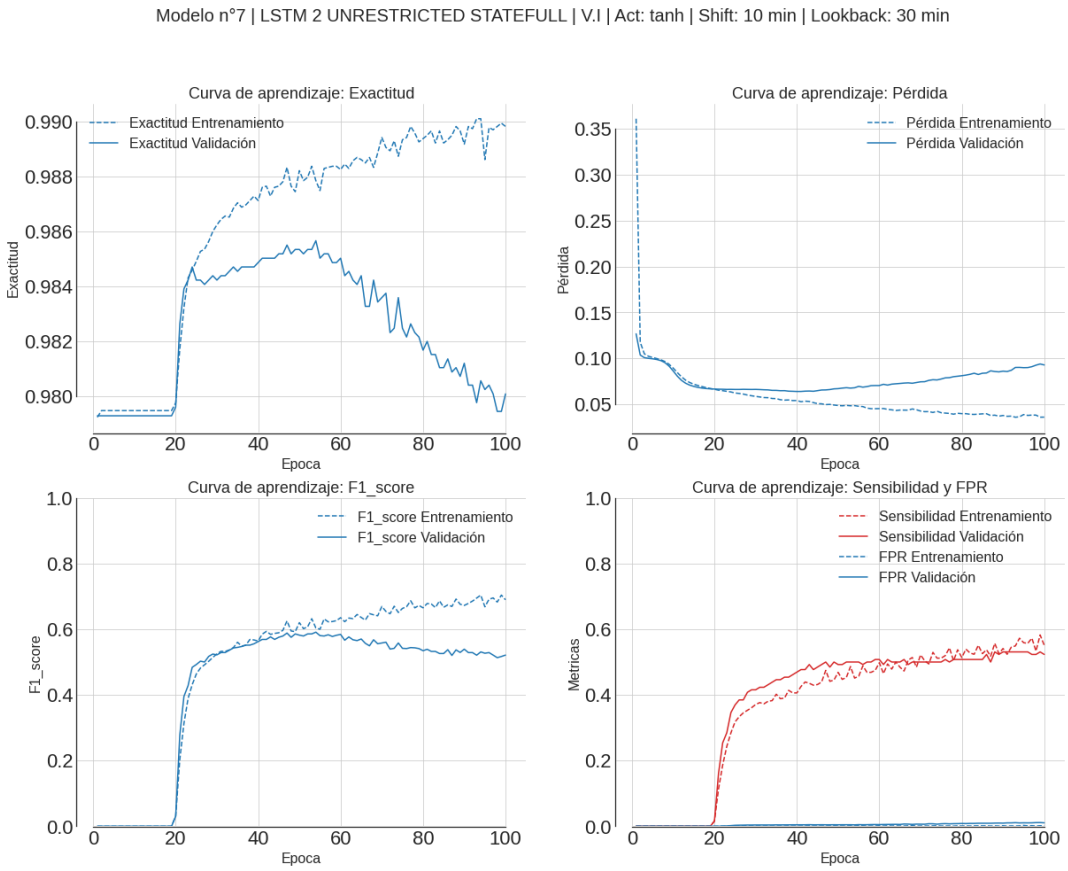


Figura 4.64: Curvas de aprendizaje caso VII modelos LSTM Iteración I.

La variante *LSTM Stateful* procesa la señal como una sola, y no toma ventanas temporales aleatorias como en los casos anteriores. Las curvas de la Figura 4.64 muestran que a partir de la época 40 aproximadamente, el modelo comienza a sobreajustarse. Los valores alcanzados en términos de métricas están un poco por debajo de los obtenidos en los casos anteriores.

Evaluación en conjunto de testeo

Finalmente, los 7 casos son evaluados en el conjunto de testeo. En la Tabla 4.28 se muestran los resultados de esta evaluación.

Tabla 4.28:Métricas en conjunto de testeo para modelos LSTM Iteración I.

Caso	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
I	0.0274	0.9925	0.7134	0.7943	0.0017
II	0.0455	0.9899	0.5414	0.6855	0.0008
III	0.0243	0.9934	0.7771	0.8271	0.0021
IV	0.0349	0.9893	0.5159	0.6612	0.0009
V	0.0313	0.9915	0.7070	0.7708	0.0026
VI	0.0295	0.9920	0.6688	0.7721	0.0013
VII	0.0682	0.9883	0.5000	0.6250	0.0019

En general de acuerdo con las curvas de aprendizaje, las diferentes variantes parecen tener un rendimiento muy similar y no hay ninguna que haya destacado por sobre las otras.

Por otro lado, en cuanto al rendimiento de los modelos en los datos de testeo, el Caso III (*Unrestricted Stateless*) exhibe el mejor desempeño en términos de métricas, alcanzando una sensibilidad de 0.78 y un F1-score de 0.83.

Iteración II

Para evaluar la influencia de otros parámetros se realiza una segunda iteración que contempla las variantes detalladas a continuación.

Se prueban diferentes combinaciones de los parámetros que se muestran a continuación.

- Dropout y Dropout recurrente: [0, 0.5]
- Cantidad de capas ocultas: [2, 3]
- Cantidad de neuronas por capa: [8, 16, 32]

Se entrenaron un total de 16 modelos, de los cuales fueron seleccionados 10 mediante visualización de sus curvas de aprendizaje (ver Anexo). Los resultados de la evaluación de los modelos seleccionados se resumen en la Tabla 4.29.

Tabla 4.29: Métricas en conjunto de testeo modelos LSTM Iteración II.

Modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0065	0.9979	0.9655	0.9540	0.0013
2	0.2558	0.9651	0.1412	0.1505	0.0164
3	0.6759	0.9429	0.1344	0.2349	0.0007
4	0.0291	0.9919	0.7679	0.8152	0.0028
5	0.0315	0.9898	0.7238	0.7683	0.0038
6	0.0274	0.9925	0.7134	0.7943	0.0017
7	0.0243	0.9934	0.7771	0.8271	0.0021
8	0.0349	0.9893	0.5159	0.6612	0.0009
9	0.0313	0.9915	0.7070	0.7708	0.0026
10	0.0295	0.9920	0.6688	0.7721	0.0013

De acuerdo con los resultados de la Tabla 4.29, de los 10 modelos seleccionados el modelo n°1 muestra el mejor rendimiento en términos de métricas, alcanza la máxima exactitud (99,8%), máxima sensibilidad (0.97) y el máximo F1-score (0.95), así como también la mínima pérdida y con una tasa de falsos positivos muy baja.

Finalmente, de los modelos evaluados, el de mejor rendimiento es uno de los modelos de la Iteración I, correspondiente a la variante *LSTM Bidirectional*. Su arquitectura corresponde a 2 capas *LSTM* de 16 y 8 unidades respectivamente, con *dropout* de 0.5 y una función de activación *tanh*. El etiquetado de muestras positivas corresponde a 10 minutos previos al evento y un tamaño de ventana de temporalización de 15 minutos.

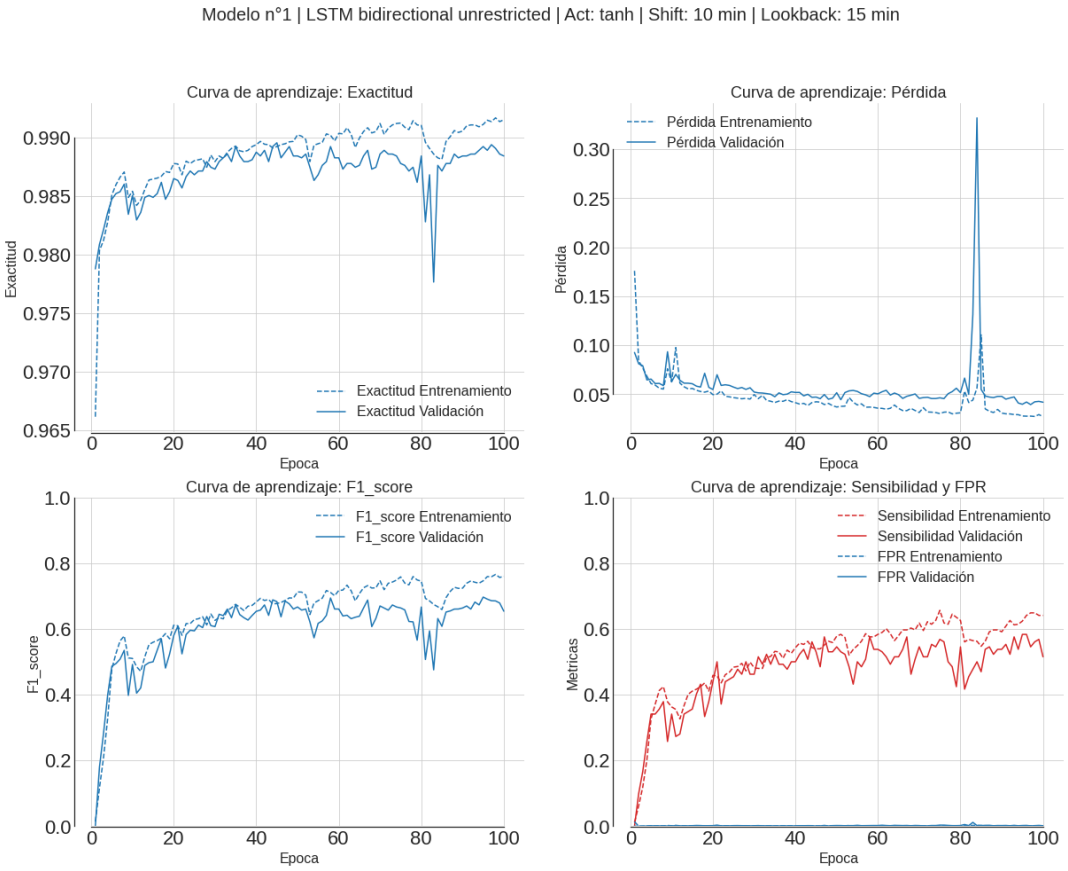


Figura 4.65: Curvas de aprendizaje del mejor modelo LSTM Iteración II.

Iteración III

Usando la misma arquitectura del mejor modelo de la iteración anterior se evalúa el efecto de variar el tamaño de la ventana de temporalización, evaluando modelos con distintos valores de este parámetro.

- Tamaño ventana temporalización (minutos): [15, 20, 60, 120, 180, 240]

Tabla 4.30: Métricas en conjunto de testeo modelos LSTM Iteración III.

Modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0332	0.9915	0.7256	0.7829	0.0028
2	0.3364	0.9924	0.7412	0.8103	0.0019
3	0.0229	0.9923	0.8011	0.8286	0.0032
4	0.0239	0.9915	0.8133	0.8036	0.0046
5	0.0231	0.9943	0.8750	0.8642	0.0032
6	0.0236	0.9913	0.8389	0.7886	0.0057

De los modelos evaluados, el n°5 destaca en términos de métricas, alcanzando la máxima exactitud (99.4%), máxima sensibilidad (0.88) y máximo F1-score (0.86).

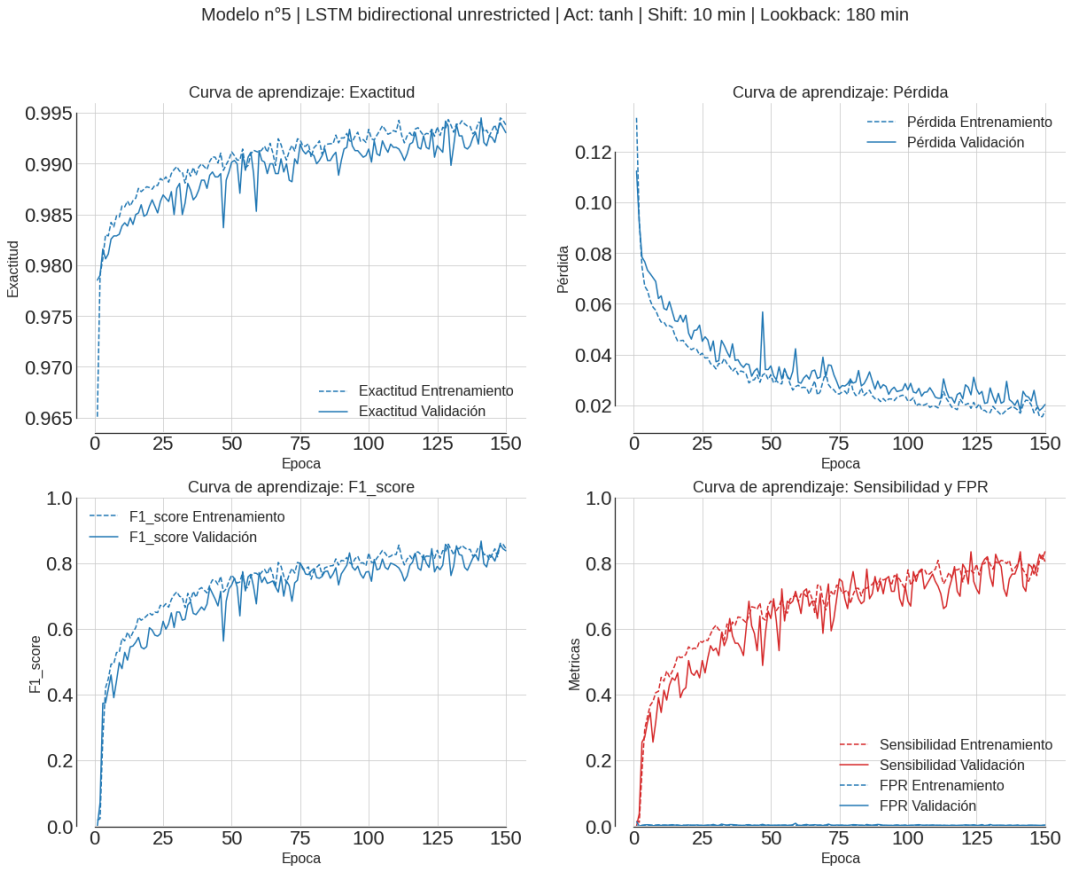


Figura 4.66: Curvas de aprendizaje mejor modelo LSTM Iteración III.

De acuerdo con lo anterior, se considera que un tamaño de ventana de 180 min optimiza los resultados y por tanto, en todos los modelos *LSTM* posteriores se utiliza este valor.

Iteración IV

En esta iteración se usa el modelo seleccionado en la iteración anterior y se evalúa el efecto de cambiar la cantidad de muestras positivas previas al evento. Los resultados obtenidos se resumen en la Tabla 4.31.

- Muestras positivas (minutos previos al evento): [10, 20, 30, 60]

Tabla 4.31: Métricas en conjunto de testeo modelos LSTM Iteración IV.

Modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0231	0.9943	0.8750	0.8642	0.0032
2	0.0287	0.9893	0.8328	0.8663	0.0039
3	0.0290	0.9894	0.9469	0.9188	0.0077
4	0.0426	0.9850	0.9782	0.9445	0.0139

De la Tabla 4.31 se observa que la pérdida aumenta muy levemente con el aumento de muestras etiquetadas como positivas reales antes de la ocurrencia del evento que se quiere predecir. Este comportamiento también se percibe con la tasa de falsos positivos, aunque en ambos casos las métricas se mantienen en valores razonables.

El F1-score y sensibilidad aumentan con el aumento de muestras etiquetadas como positivas reales, lo que indica que el modelo mejora su rendimiento en la medida que más se quiere anticipar la predicción de un evento de alto torque. Sin embargo, este comportamiento puede explicarse por lo siguiente: El hecho de aumentar las muestras positivas, y en consecuencia disminuir las negativas, genera que las clases estén más balanceadas y por tanto se mejora aparentemente el desempeño del modelo en términos de la sensibilidad.

De todas formas, los modelos de la Iteración IV corresponden a los modelos recurrentes definitivos de la Versión I.

A continuación, se muestran los resultados del modelo n°4 que alcanzó las mejores métricas de sensibilidad (0.98) y F1-score (0.94).

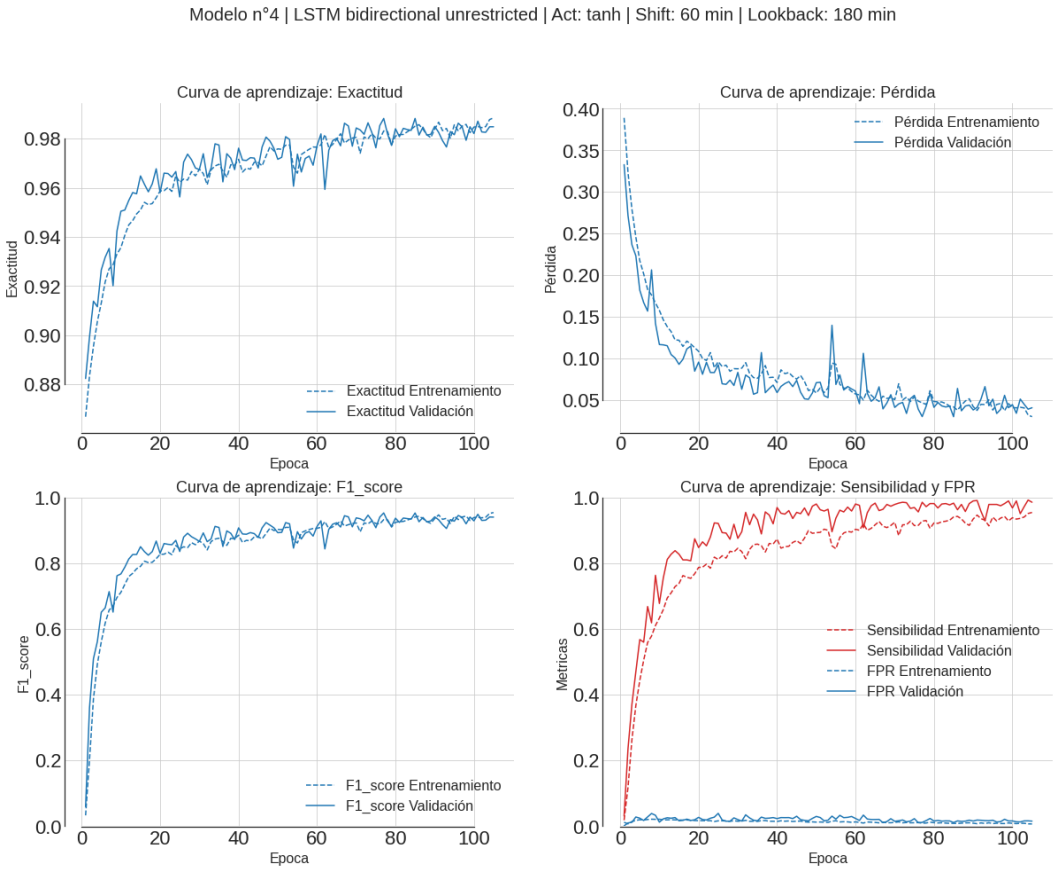


Figura 4.67: Curvas de aprendizaje modelo n°4 LSTM Iteración IV.

Las curvas de aprendizaje mostradas en la Figura 4.68 muestran resultados excepcionales en cuanto a métricas. La pérdida es decreciente tanto en validación como entrenamiento y no muestra indicios de sobreajuste. Por su parte la sensibilidad y F1-score tienden a valores muy cercanos a 1.0 con una tasa de falsos positivos bien controlada en torno a 0.

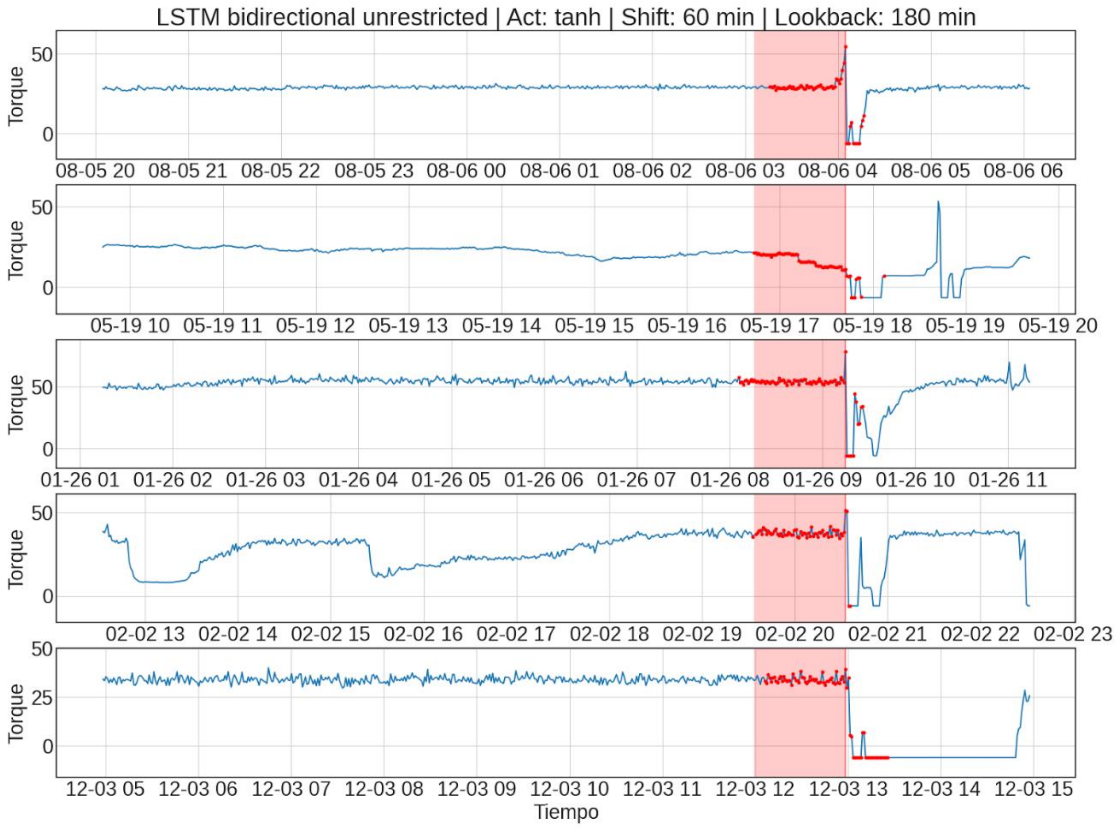


Figura 4.68: Señales temporales de algunos eventos evaluadas en modelo definitivo LSTM Versión I.

La Figura 4.69 muestra el desempeño del modelo en algunas señales que contienen algunos de los 81 eventos utilizados para el desarrollo de los modelos. La franja roja indica los minutos previos al evento que se etiquetan como precursores (muestras positivas reales), mientras que los puntos rojos indican las muestras clasificadas positivas por el modelo (muestras positivas predichas).

Los resultados de la Figura 4.69 confirman el buen rendimiento del modelo, donde se observa una baja cantidad de falsos positivos y, a diferencia de los modelos *MLP*, logra predecir el alto torque con anticipación.

Sin embargo, al evaluar con señales temporales que no son parte del conjunto de datos con que fueron desarrollados los modelos (Figura 4.69), el desempeño del modelo ya no es tan excepcional, poniendo en evidencia la baja capacidad de generalización del modelo, indicando que el modelo estaría sobreajustado. Más adelante, en el final de esta sección, se retoma este punto con mayor detalle.

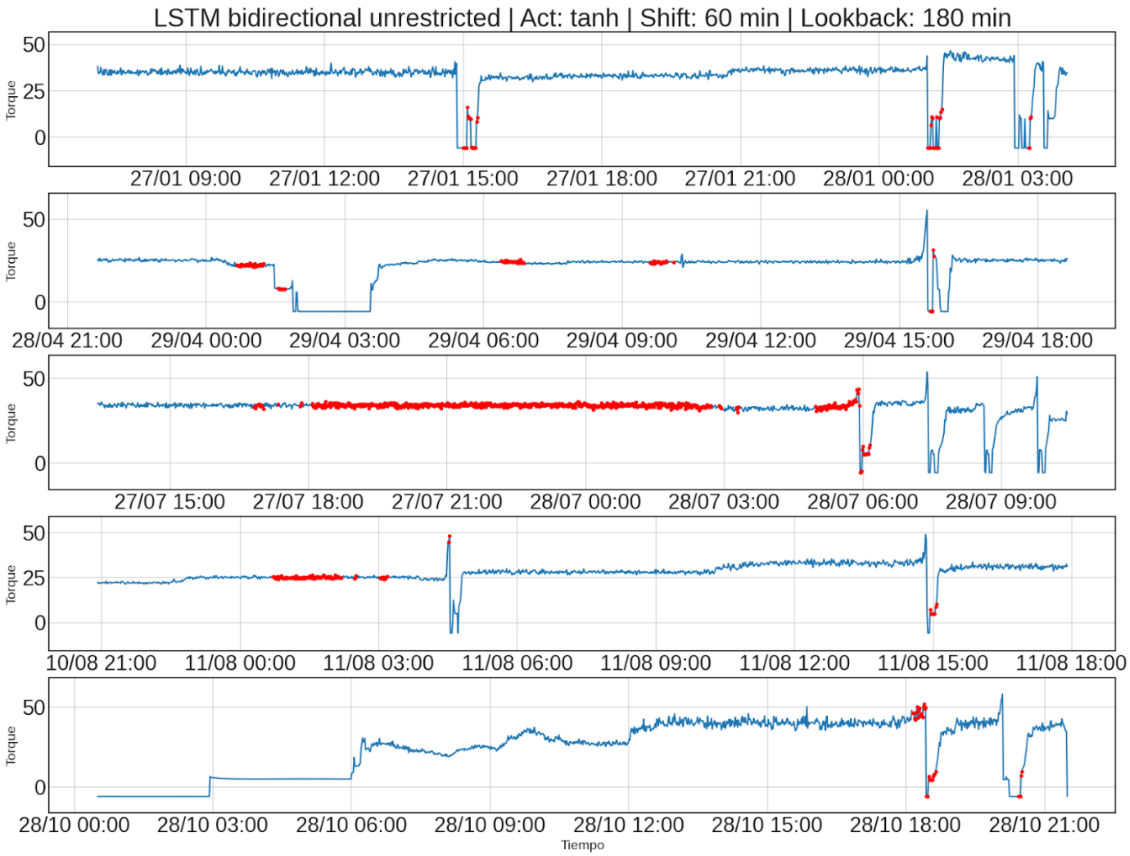


Figura 4.69: Señales temporales evaluadas en modelo definitivo LSTM Versión I.

Modelos con capas Convolucionales

De forma análoga a los modelos recurrentes se desarrollaron modelos con capas convolucionales.

Iteración I

Las características de los modelos convolucionales en esta primera iteración son:

- Capa de entrada: Tiene tantas unidades como parámetros (18).
- Capa oculta 1: 1 capa *Conv1D* de 9 filtros tamaño 3.
- Capa oculta 2: 1 capa *MaxPool* de *pool_size* 3.
- Capa oculta 3: 1 capa *Flatten*.
- Capa oculta 4: 1 capa *Dense* de 8 unidades.
- Capa de salida: Una sola unidad con activación Sigmoide.

Para el entrenamiento se utilizan los siguientes parámetros:

- Tamaño de batch: 128
- Épocas: 100
- Optimizador: Adam
- Learning rate: 0.001
- Función de pérdida: Binary Cross Entropy
- Muestras etiquetadas positivas: 10 minutos
- Tamaño ventana temporalización: 30 minutos

Las arquitecturas de los casos aquí expuestos se pueden encontrar en los Anexos, en esta sección solamente se muestran las curvas de aprendizaje de cada uno. La Tabla 4.32 resume la arquitectura utilizada en cada caso.

Tabla 4.32: Casos de estudio modelos CNN Iteración I.

Caso	activación	Capas Conv (Filtros/Kernel)	Capas MaxPool (Pool)	Dropout	Capas Densas
I	relu	Conv1d (9/3)	1D (3)	0.0	(8)
II	tanh	Conv1d (9/3)	1D (3)	0.0	(8)
III	relu	Conv1d (9/3)	1D (3)	0.5	(8)
IV	relu	Conv2d (9/3)	2D (3)	0.0	(8)
V	relu	Conv2d (9/(3,4))	2D (3)	0.0	(8)
VI	relu	Conv1D (9/3)(9/1)	1D (3)	0.5	(8)

Caso I

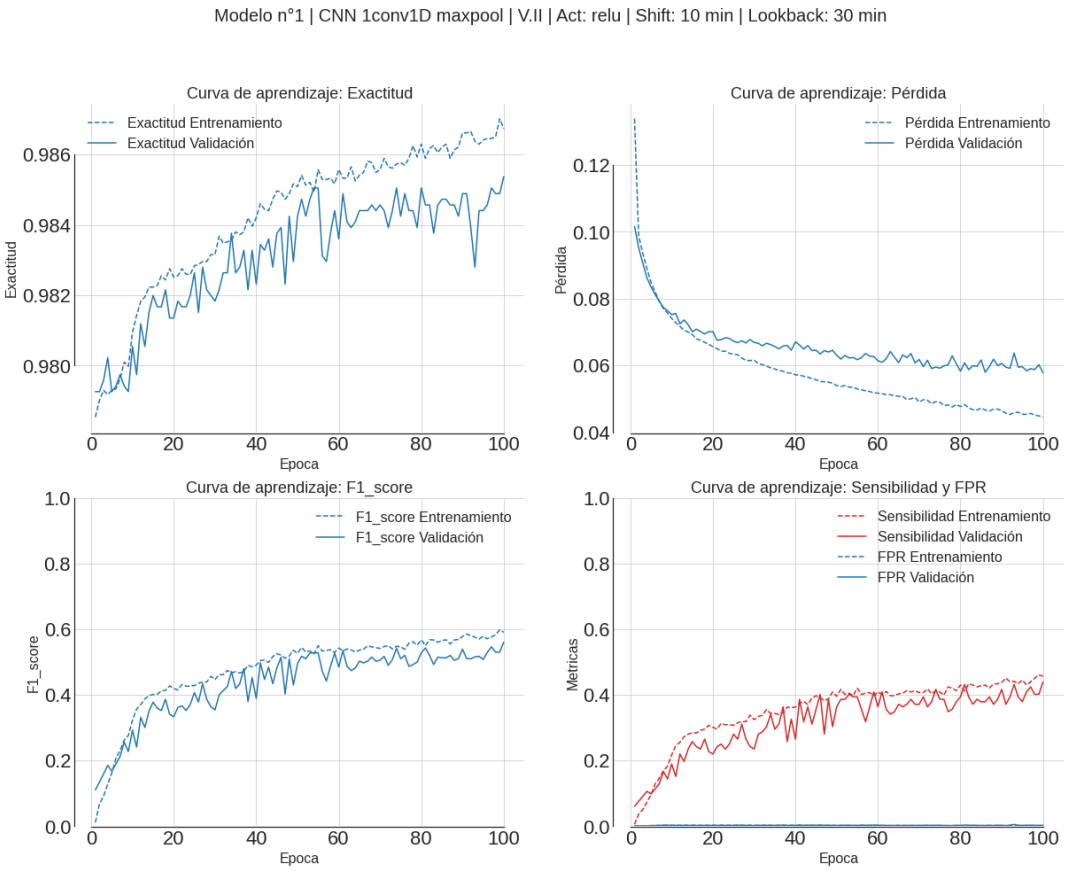


Figura 4.70: Curvas de aprendizaje Caso I modelos CNN Iteración I.

Como se observa en la Figura 4.71 la pérdida es decreciente tanto en validación como entrenamiento indicando convergencia. Por otro lado, la sensibilidad y F1-score muestran resultados similares a los obtenidos con modelos *MLP*.

Caso II

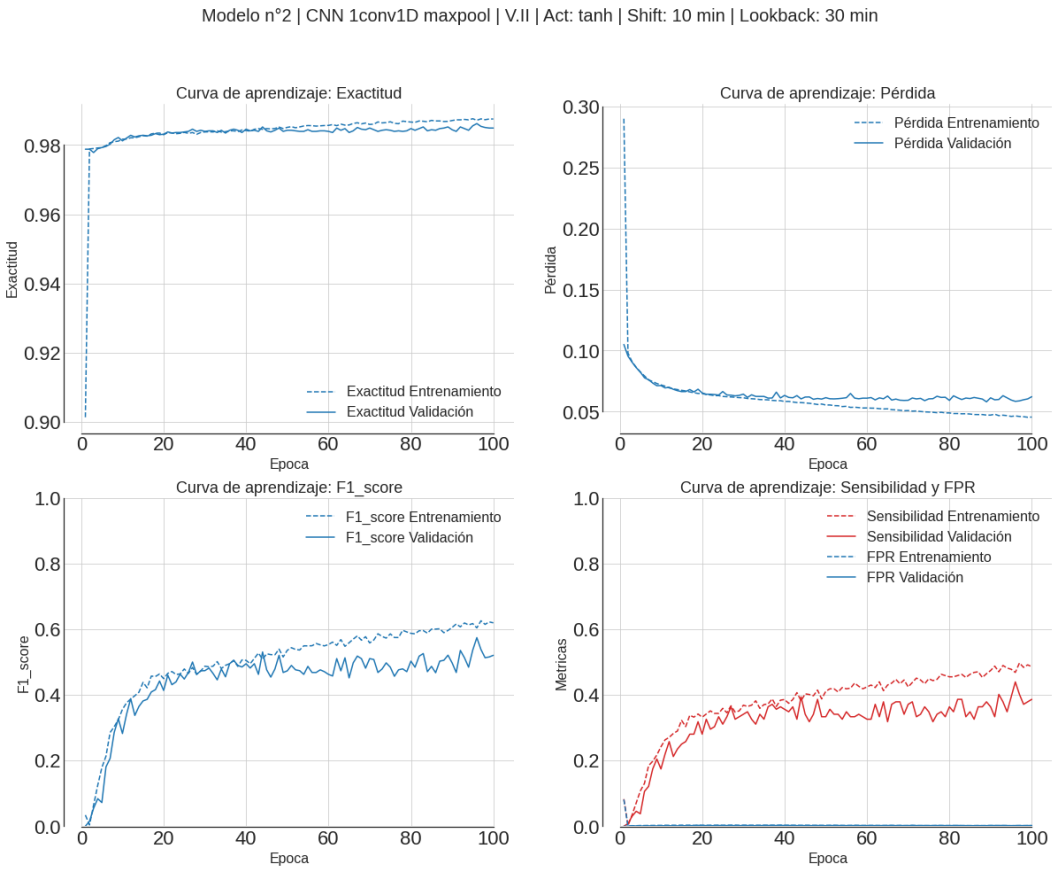


Figura 4.71: Curvas de aprendizaje Caso II modelos CNN Iteración I.

Idéntico al Caso I pero con activación *tanh*. Las curvas de la Figura 4.72 muestran resultados bastante similares.

Caso III

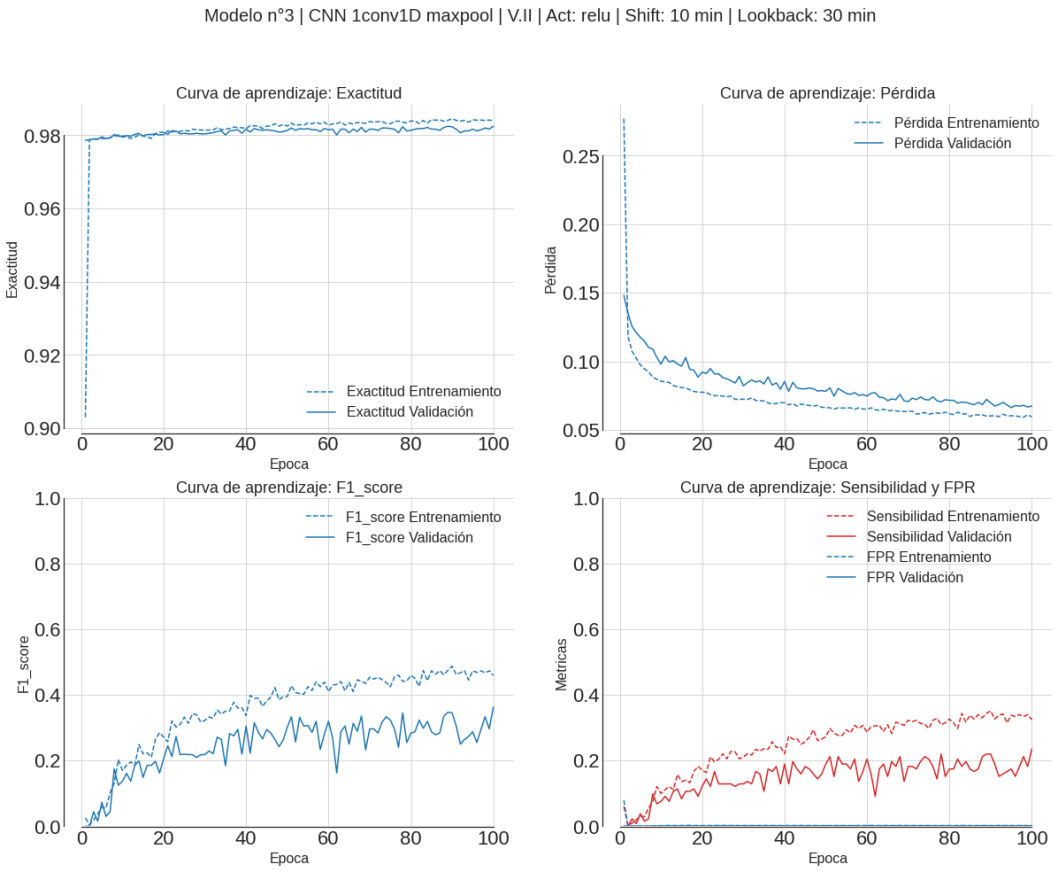


Figura 4.72 Curvas de aprendizaje Caso III modelos CNN Iteración I.

El Caso I mostraba indicios de sobreajuste en la curva de pérdida. Como se observa en la Figura 4.73, añadir regularización por *dropout* (0.5) elimina el indicio de sobreajuste en la pérdida de validación, aunque las curvas de sensibilidad y F1-score muestran realmente una baja en el rendimiento del modelo.

Caso IV

En este caso los datos son re-estructurados como muestra la Figura 4.74, similar a la forma en que son procesadas las imágenes.

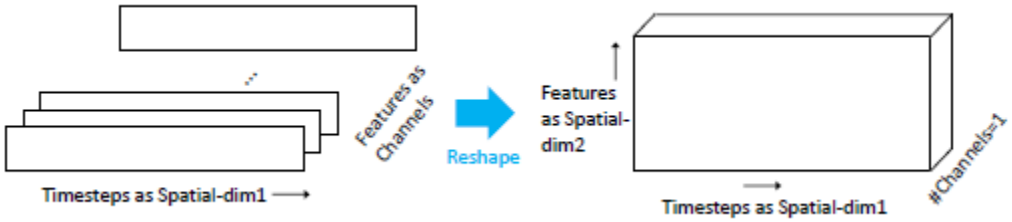


Figura 4.73: Re-estructuración de datos desde un tensor 2D a un tensor 3D.

Este modelo en realidad es equivalente al Caso I, de hecho, las curvas de aprendizaje mostradas en la Figura 4.75 son similares a las de la Figura 4.71.

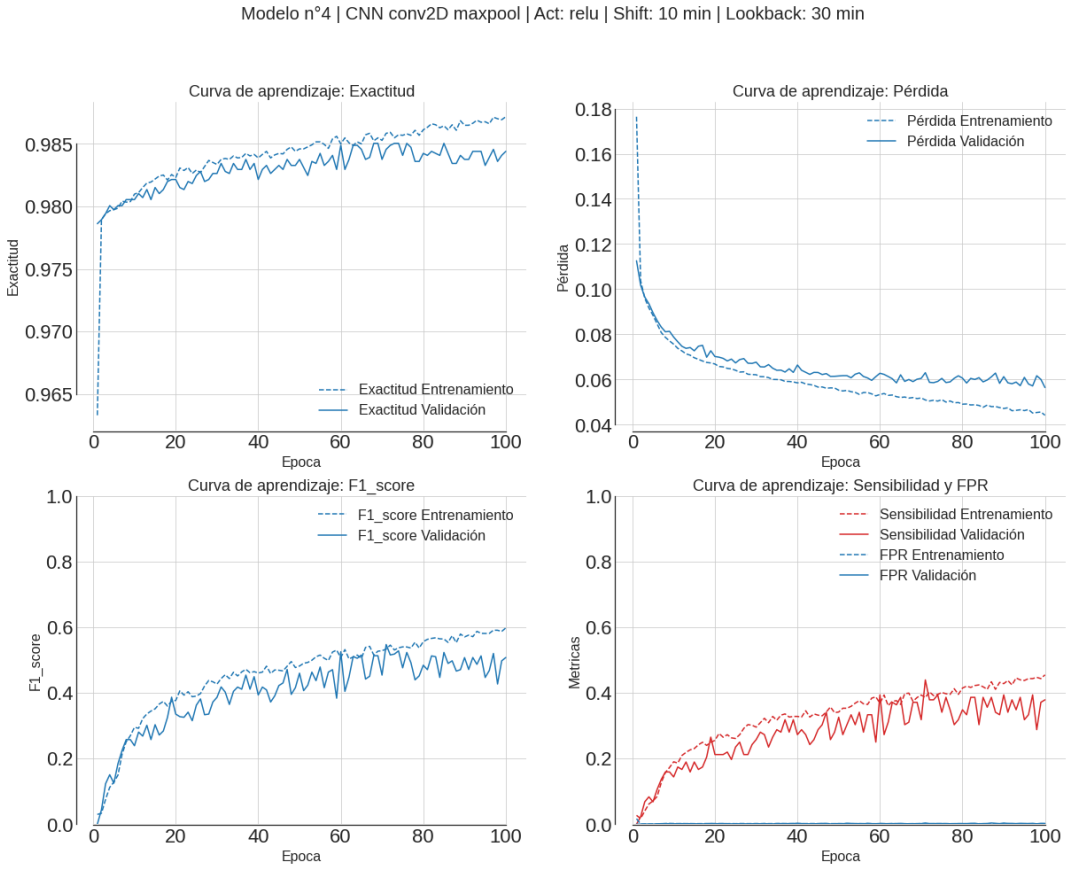


Figura 4.74: Curvas de aprendizaje Caso IV modelos CNN Iteración I.

Caso V

Este modelo es construido para aprender patrones tanto temporales como espaciales entre variables que podrían estar relacionadas localmente, para ellos se utiliza la misma estructura de datos del Caso IV, pero con un kernel de tamaño (3,4), que permite localizar patrones en un espacio temporal de 3 muestras consecutivas en 4 variables a la vez.

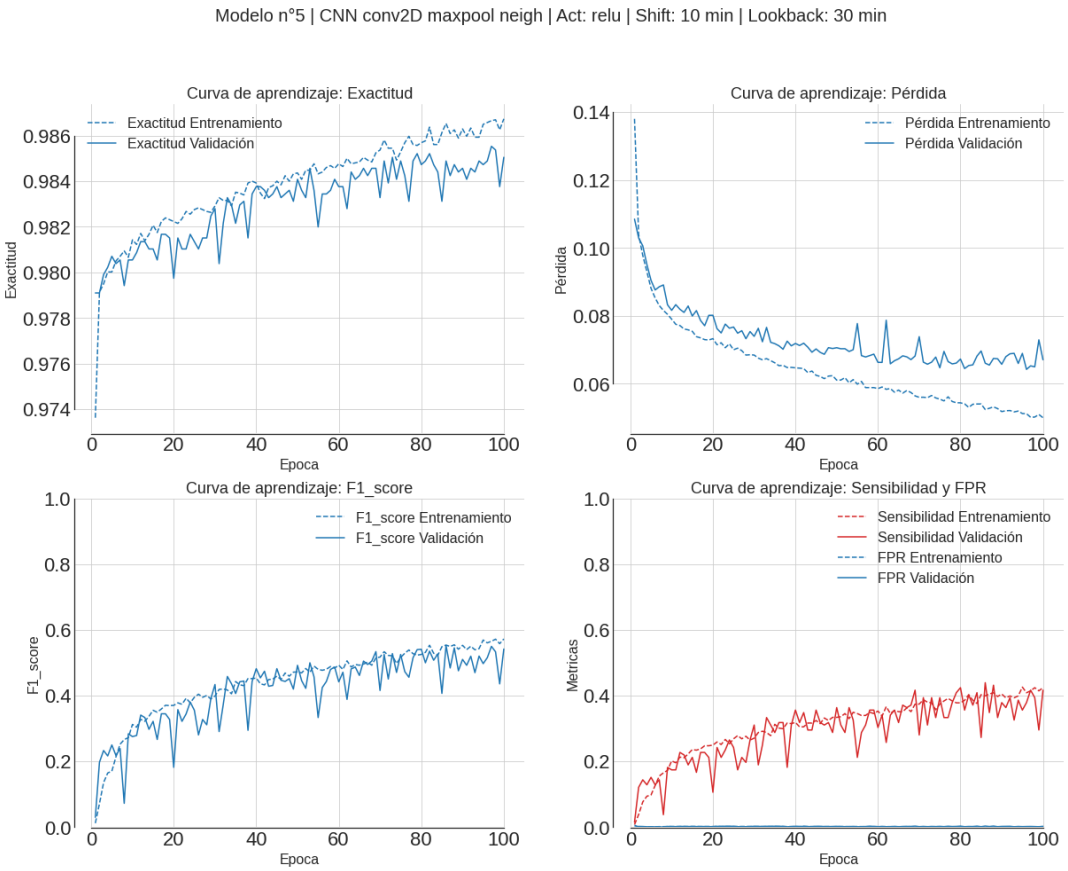


Figura 4.75: Curvas de aprendizaje Caso V modelos CNN Iteración I.

El modelo entrega resultados similares a las otras variantes como se puede ver en la Figura 4.76, sin mejoras considerables.

Caso VI

Esta variante tiene 1 capa Conv1D con 18 filtros de tamaño 3, y una capa Conv1x1 con 9 filtros de tamaño 1, cuyo objetivo es comprimir los canales en 1 solo.

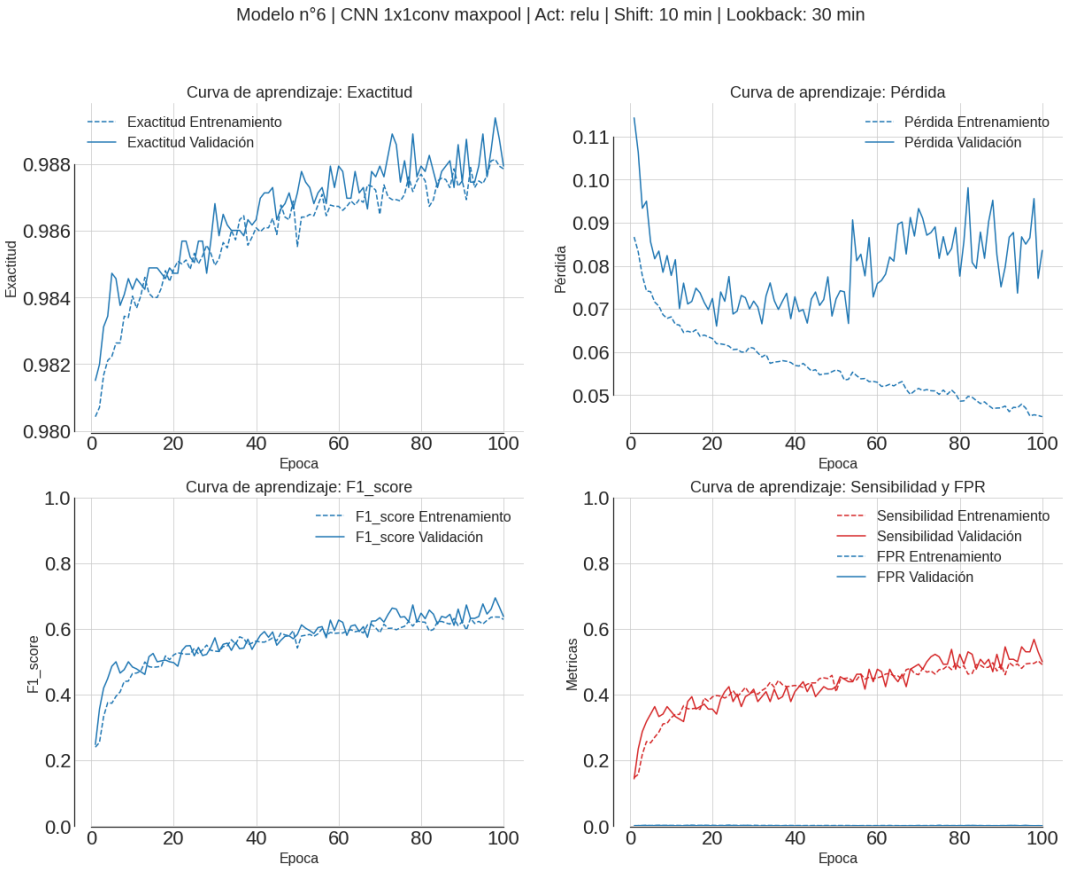


Figura 4.76: Curvas de aprendizaje Caso I modelos CNN Iteración I.

Las curvas de aprendizaje de la Figura 4.77 muestran un mejor rendimiento que en las variantes anteriores, aunque en la curva de pérdida se aprecia claramente que a partir de la época 40 el modelo comienza a sobreajustarse.

La Tabla 4.33 resume los resultados de evaluación de los casos desarrollados en la Iteración I. Los Casos I y VI son los que muestran un rendimiento más equilibrado en términos de sensibilidad y F1-score con valores en torno a 0.5 en ambos casos.

Estos resultados aún están lejos de lo obtenido con los modelos recurrentes, sin embargo, en las siguientes iteraciones se logra mejorar el desempeño.

Tabla 4.33: Métricas en conjunto de testeo modelos CNN Iteración I.

Caso	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
I	0.0590	0.9848	0.4459	0.5426	0.0041
II	0.2031	0.9099	0.5541	0.1991	0.0827
III	0.0642	0.9843	0.2611	0.4019	0.0008
IV	0.0642	0.9843	0.2611	0.4019	0.0008
V	0.0628	0.9859	0.4076	0.5401	0.0021
VI	0.0883	0.9878	0.4968	0.6215	0.0021

Iteración II

Se prueban diferentes combinaciones de los siguientes parámetros.

- Dropout: [0, 0.5]
- Cantidad de capas Conv: [1, 2, 3]
- Cantidad de filtros por capa: [4, 6, 8, 9, 10, 12, 16, 18, 32]
- Tamaño del kernel: [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 32]
- Cantidad de capas MaxPool: [0, 1, 2]
- Tamaño Pool: [1, 2, 3, 4, 6, 8]
- Cantidad de capas Densas: [1, 2]
- Cantidad de neuronas (Densa): [8, 16, 32, 64]

Se entrenaron un total de 47 modelos de los cuales solamente 5 fueron seleccionados a partir del análisis de las curvas de aprendizaje (ver Anexo). En la Tabla 4.34 se muestran los resultados de evaluación de los modelos en el conjunto de testeo.

Tabla 4.34: Métricas en conjunto de testeo modelos CNN Iteración II.

modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0279	0.9928	0.8313	0.8313	0.0037
2	0.0344	0.9893	0.5528	0.6819	0.0014
3	0.0283	0.9933	0.7391	0.8207	0.0013
4	0.0208	0.9948	0.8819	0.8765	0.0028
5	0.0177	0.9952	0.8323	0.8787	0.0013

De acuerdo con la Tabla 4.34 el modelo n°5 es el de mejor rendimiento en cuanto a métricas, con una pérdida y FPR mínimos, además de obtener la máxima exactitud y F1-score, junto con el segundo valor de sensibilidad más alto.

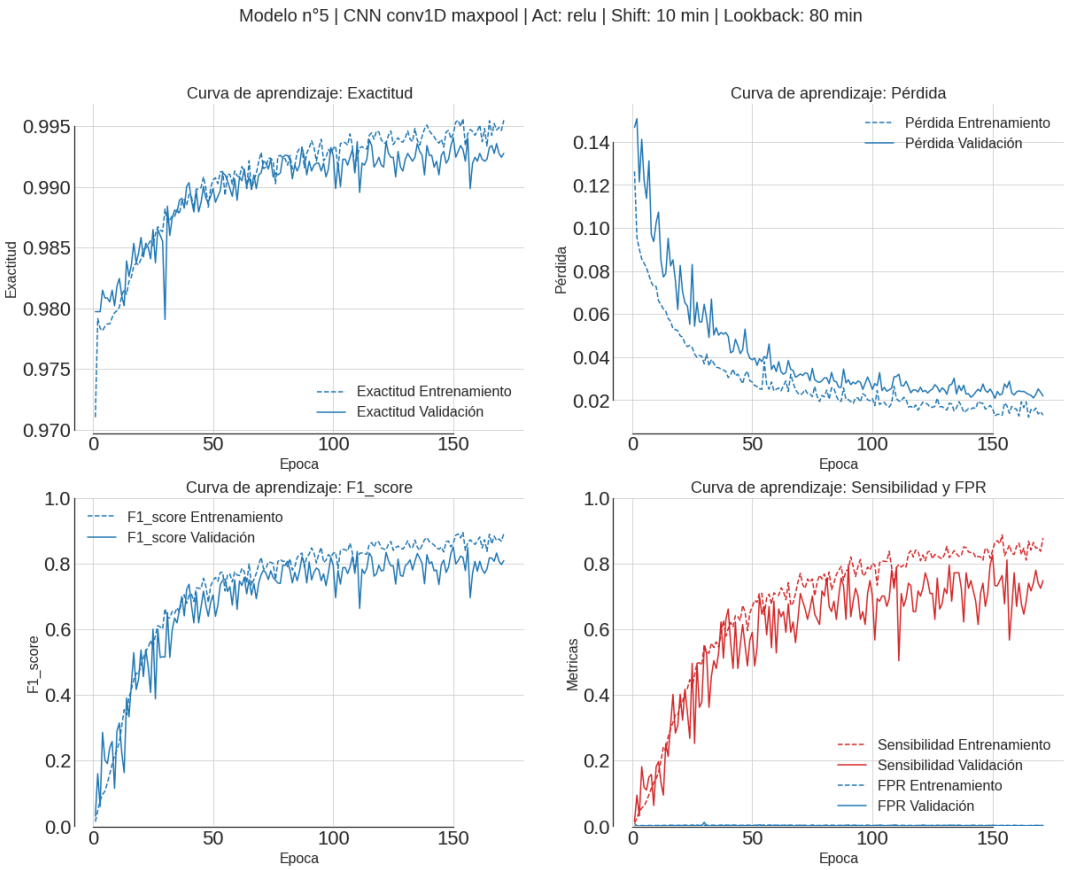


Figura 4.77: Curvas de aprendizaje del mejor modelo CNN Iteración II.

Las curvas de aprendizaje del modelo seleccionado (Figura 4.78) muestran una notable mejora en el rendimiento del modelo en comparación con la iteración anterior, que se acercan a los resultados vistos con modelos recurrentes.

La Figura 4.79 muestra la arquitectura del modelo seleccionado, donde se utilizó una función de activación *relu*, con una capa convolucional de 16 filtros con tamaño de kernel 20, con *dropout* de 0.5 a la salida, una capa *Maxpooling* con un *pool_size* 4 y una capa *dense* de 32 unidades. El etiquetado de muestras positivas reales es de 10 minutos previos al evento y el tamaño de ventana de 80 minutos.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_6 (Conv1D)           (None, 61, 16)            5776

 dropout_5 (Dropout)         (None, 61, 16)            0

 max_pooling1d_6 (MaxPooling  (None, 6, 16)            0
 1D)

 flatten_6 (Flatten)         (None, 96)                0

 dense_8 (Dense)             (None, 32)                3104

 output (Dense)              (None, 1)                 33

=================================================================
Total params: 8,913
Trainable params: 8,913
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.78: Arquitectura mejor modelo CNN Iteración II.

Iteración III

Se utiliza la arquitectura del mejor modelo de la iteración anterior (Figura 4.79) y se evalúa con diferentes tamaños de ventana para la temporalización.

- Tamaño de ventana: [60, 120, 240, 360]

Los resultados de esta iteración son mostrados en la Tabla 4.35, donde se observa que el modelo n°3 asociado a un tamaño de ventana de 240 minutos, maximiza la exactitud, la sensibilidad y el F1-score.

Tabla 4.35: Métricas en conjunto testeo modelos CNN Iteración III.

modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0204	0.9928	0.7293	0.8250	0.0009
2	0.0203	0.9929	0.7409	0.8173	0.0016
3	0.0175	0.9948	0.9396	0.8750	0.0041
4	0.0173	0.9935	0.7670	0.8438	0.0012

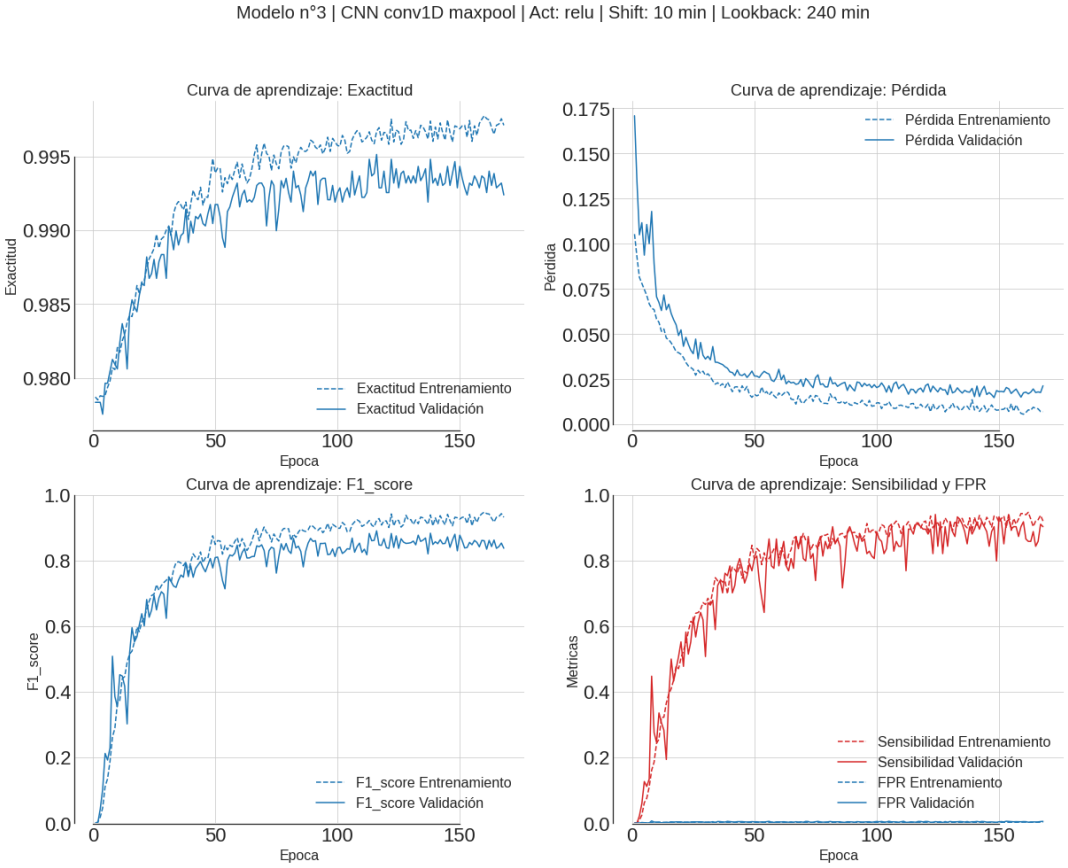


Figura 4.79: Curvas de aprendizaje mejor modelo CNN Iteración III.

Las curvas de aprendizaje del modelo de mejor rendimiento son mostradas en la Figura 4.80. Los resultados son sensiblemente mejores que los obtenidos en la iteración

anterior, por tanto, de ahora en adelante todos los modelos Convolucionales utilizan un tamaño de ventana de 240 minutos.

Iteración IV

Se utiliza la arquitectura y parámetros del mejor modelo de la iteración anterior y se evalúa el efecto de cambiar la cantidad de muestras etiquetadas como positivas reales.

- Muestras positivas (minutos previos al evento): [10, 20, 30, 60]

Los resultados de esta iteración se resumen en la Tabla 4.36. Se observa que la métrica F1-score aumenta con la cantidad de muestras positivas reales. No se aprecian relaciones de esta naturaleza en las otras métricas.

Tabla 4.36: Métricas en conjunto testeo modelos CNN iteración IV.

modelo	Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1	0.0175	0.9948	0.9396	0.8750	0.0041
2	0.0325	0.9931	0.8841	0.9097	0.0024
3	0.0234	0.9939	0.9350	0.9499	0.0022
4	0.0388	0.9909	0.9706	0.9636	0.0062

En base a lo anterior, los resultados se maximizan para un número de muestras positivas reales equivalente a 60 minutos previos al evento. Sin embargo, como se mencionó previamente esto puede explicarse por el balance de las clases, aunque también puede deberse a la presencia de precursores en la hora previa al evento de alto torque.

De todas formas, los modelos de la iteración IV son considerados los modelos Convolucionales definitivos de la Versión I.

La Figura 4.81 muestra las curvas de aprendizaje del modelo n°4 de la iteración IV. Nuevamente se observan resultados especialmente buenos, con las curvas de sensibilidad y F1-score llegando a valores cercanos a 1 tanto en conjuntos de entrenamiento como también en validación y testeo.

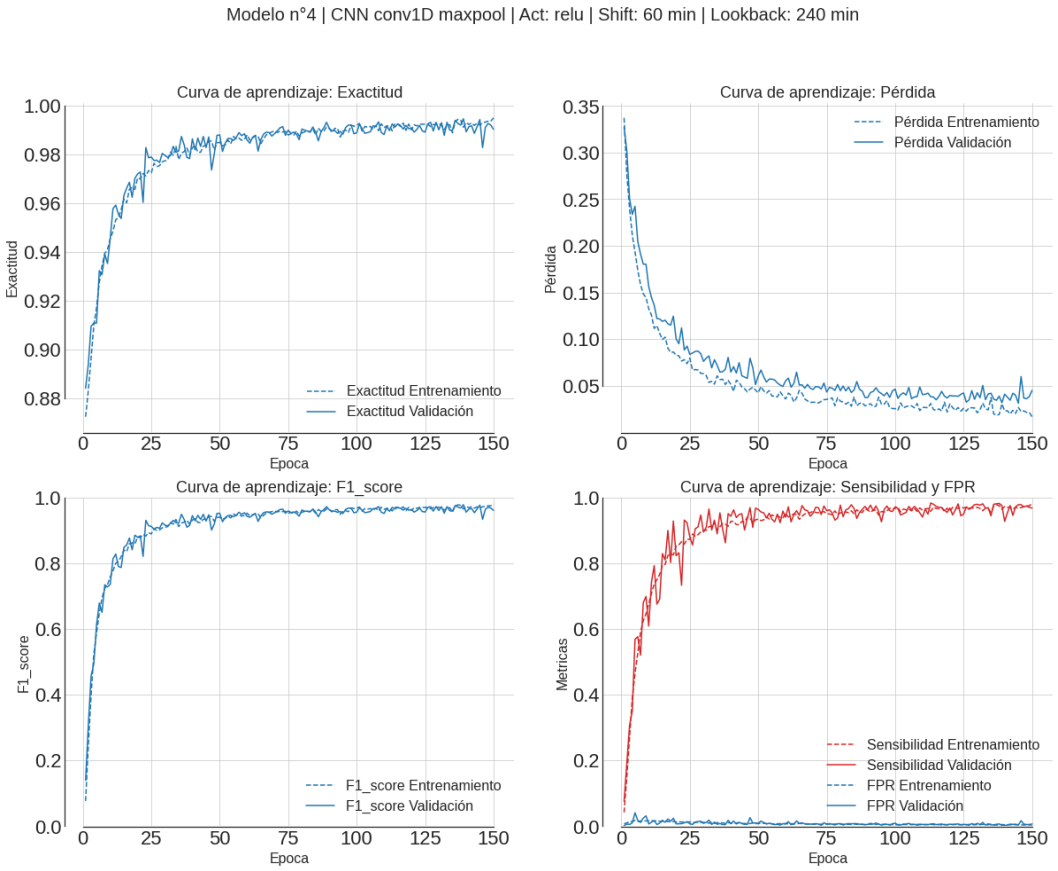


Figura 4.80: Curvas de aprendizaje

La Figura 4.82 confirma el buen rendimiento del modelo al evaluarlo en señales del conjunto de datos con que fue desarrollado el modelo, donde incluso en algunos casos el modelo se anticipa más de lo impuesto por el etiquetado previo al evento (franja roja).

Las muestras positivas predichas que están fuera de la franja roja (positivos reales) son en efecto falsos positivos, sin embargo, en este caso particular, es deseable que existan falsos positivos de esta naturaleza, siempre que sean contiguos a positivos reales, ya que serían un indicador de que los eventos de alto torque se pueden anticipar minutos antes de lo previsto.

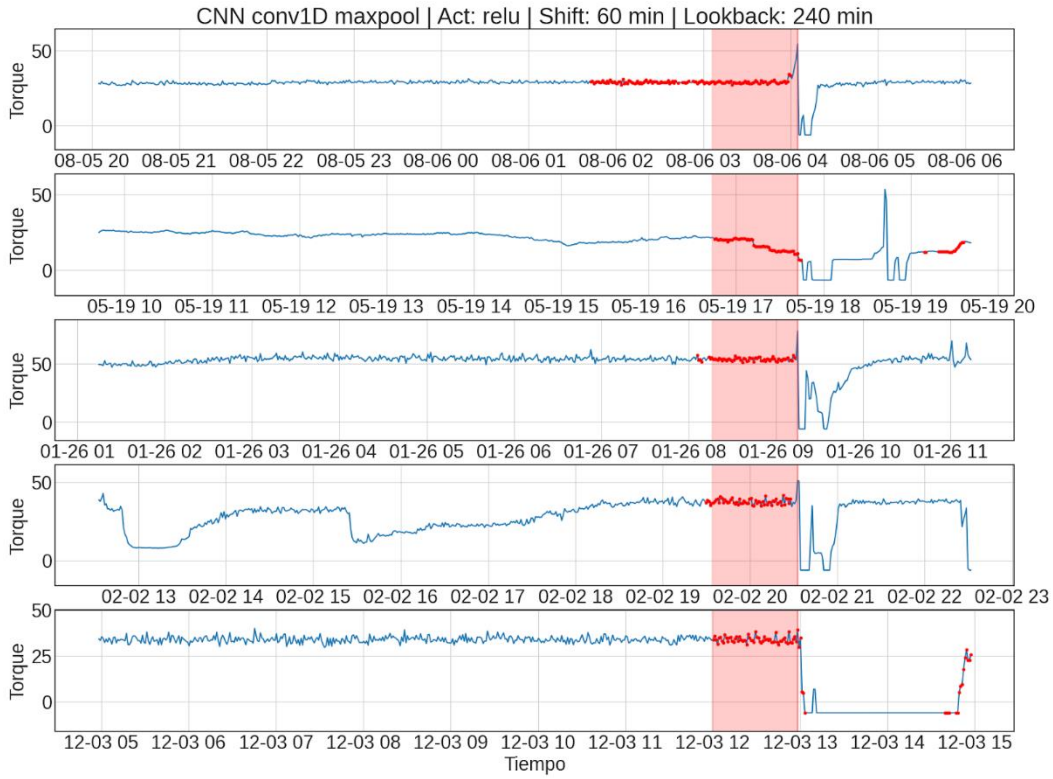


Figura 4.81: Señales temporales de algunos eventos evaluadas en modelo definitivo CNN Versión I.

Sin embargo, al igual que con el modelo recurrente definitivo, al evaluar con señales temporales totalmente nuevas (Figura 4.83), el modelo ya no tiene tan buen desempeño predictivo, evidenciando la baja capacidad de generalización del modelo.

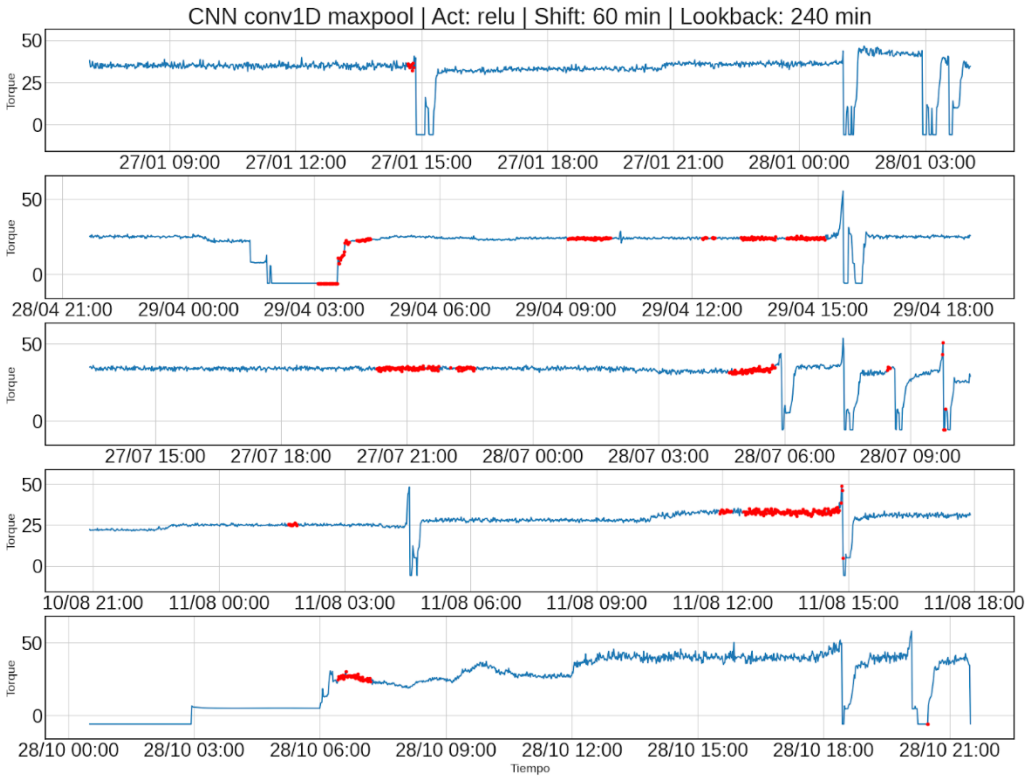


Figura 4.82: Señales temporales evaluadas en modelo definitivo CNN Versión I.

Conclusiones preliminares

Los modelos temporales de la Versión I muestran en teoría buenos resultados, tanto al observar las curvas de aprendizaje como las métricas calculadas en el conjunto de Testeo.

Sin embargo, al evaluar los modelos en señales del dominio del tiempo con datos que el modelo no conoce, el buen rendimiento no se ve reflejado en lo absoluto. Esto indica que el modelo está sobreajustado, es decir, sólo tiene un buen rendimiento en las señales de los 81 eventos de alto torque con que fueron desarrollados los modelos.

Las curvas de aprendizaje y las métricas deberían evidenciar el sobreajuste de un modelo, mediante un pobre desempeño tanto en los conjuntos de validación y testeo, sin embargo, esto no se percibe en los resultados de los modelos definitivos.

Lo anterior puede ser explicado de la siguiente forma. La temporalización de los datos genera ventanas temporales traslapadas que contienen una serie de muestras, esto implica que cada muestra está contenida en tantas ventanas como el tamaño de esa ventana, aunque en una posición levemente desplazada en el tiempo. Debido a que la temporalización se lleva a cabo antes de separar los datos en conjuntos de Entrenamiento, Validación y Testeo, hay una alta probabilidad de que una misma muestra esté presente en los 3 conjuntos de datos. lo cual implica que los 3 conjuntos contienen prácticamente la misma información, explicando

así el buen ajuste de los modelos a los datos de entrenamiento y el buen rendimiento al evaluar los conjuntos de validación y testeo. Este efecto parece no ser considerado en los resultados reportados en [22], donde se califica el desempeño de un modelo sólo en función de métricas al evaluar conjuntos de entrenamiento, validación y testeo que, tal como se explicó, aparentan ser disjuntos pero que en el fondo contienen prácticamente la misma información, y no en función de métricas calculadas usando datos nunca vistos, ni en base a una evaluación de los resultados obtenidos en el dominio del tiempo.

4.3.8. Modelos con capas recurrentes y convolucionales: Versión II.

En virtud de lo anterior, se desarrolló una segunda Versión de modelos Recurrentes y Convolucionales con la finalidad de revertir el problema del sobreajuste. Para ello los datos fueron procesados de la siguiente forma.

1. Reestructuración de datos en una única señal.
2. Etiquetado de los datos.
3. Disminuir frecuencia de muestreo.
4. Dividir datos en orden temporal en conjuntos de Entrenamiento, Validación y Testeo.
5. Temporalizar conjuntos de datos por separado.
6. Normalizar datos a partir de datos de Entrenamiento.

Modelos con capas Recurrentes

Se implementa el modelo de mejor rendimiento de la Versión I, con la misma arquitectura y parámetros, pero con los datos procesados de la que sería la forma correcta.

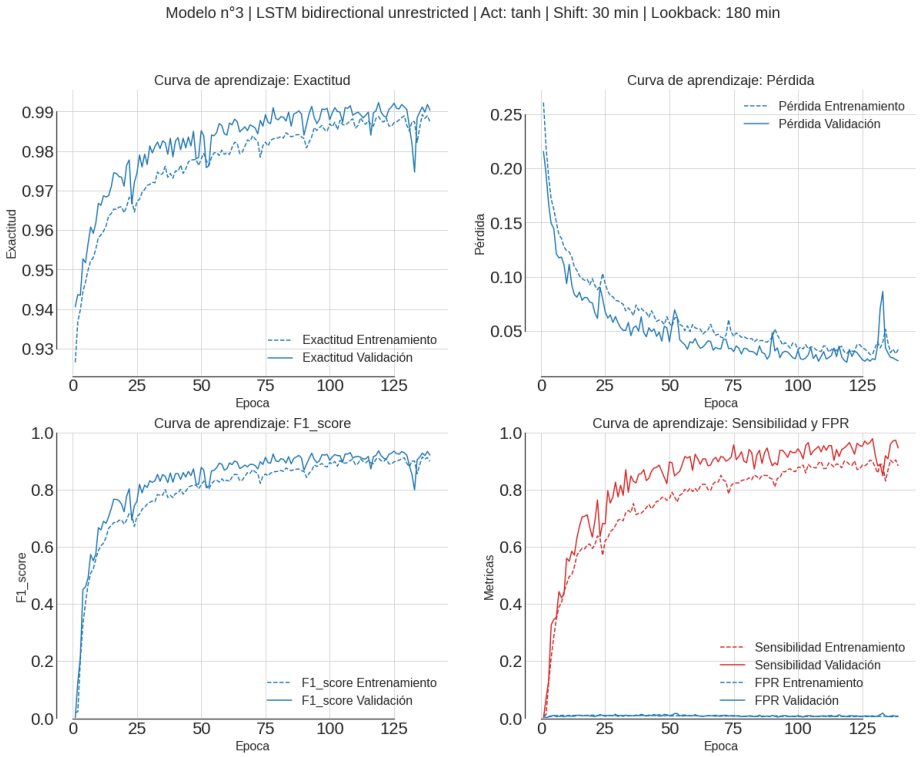


Figura 4.83: Modelo definitivo LSTM Versión I.

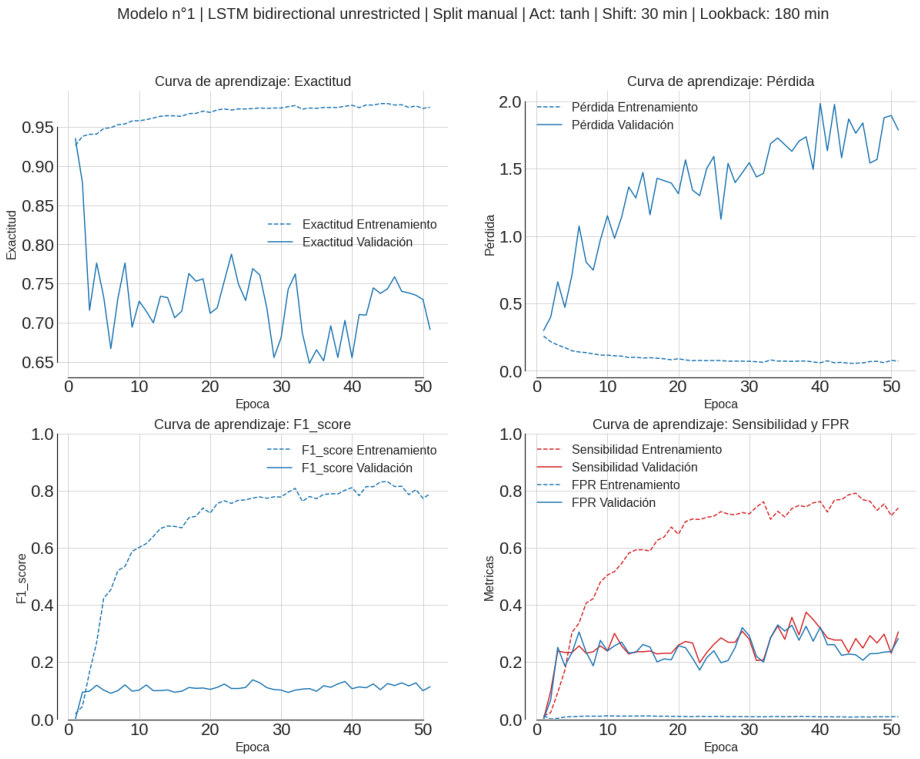


Figura 4.84: Modelo LSTM Versión II con la misma arquitectura que modelo definitivo LSTM Versión I.

Al comparar las curvas de aprendizaje del modelo recurrente definitivo de la Versión I (Figura 4.84) con su contraparte en la Versión II (Figura 4.85), notamos claramente el sobreajuste del modelo, esto se aprecia especialmente en la curva de pérdida del conjunto de validación, la cual tiene un comportamiento creciente.

Tabla 4.37: Métricas conjunto de testeo Modelo LSTM Versión II.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
1.3697	0.5928	0.4104	0.1138	0.3948

Las métricas en el conjunto de testeo tampoco son buenas, lo que termina por confirmar el sobreajuste el modelo y su baja capacidad de generalización.

Para reducir el sobreajuste se exploraron diferentes métodos, entre ellos:

- Reducir número de capas y neuronas
- Reducir tamaño de *batch*
- Agregar regularización L1, L2 y *dropout*

Se entrenaron un total de 12 modelos implementando las medidas anteriores y con los siguientes parámetros.

- Tamaño de ventana temporal: 180 min
- Muestras positivas (minutos previos al evento): 30 min
- Optimizador: Adam
- Learning rate: 0.001

La mayoría de los modelos entrenados muestran el mismo comportamiento que se tenía antes de implementar estas medidas (Figura 4.85), y solamente 4 modelos mostraron curvas de pérdida de validación de carácter decreciente (ver Anexo).

La regularización L1 y L2 junto con la simplificación de los modelos (una sola capa con menos unidades), parece dar resultados en términos de reducción de sobreajuste, o al menos tener una pérdida de validación descendente.

La curva de aprendizaje de exactitud muestra un correcto aprendizaje del modelo tanto en conjuntos de entrenamiento como validación (Figura 4.86). Sin embargo, la baja

sensibilidad (0.2) y F1-score (0.3) acusan que el modelo tiene problemas para clasificar tanto las muestras positivas como las negativas, en otras palabras, no es capaz de anticipar los eventos de alto torque con certeza.

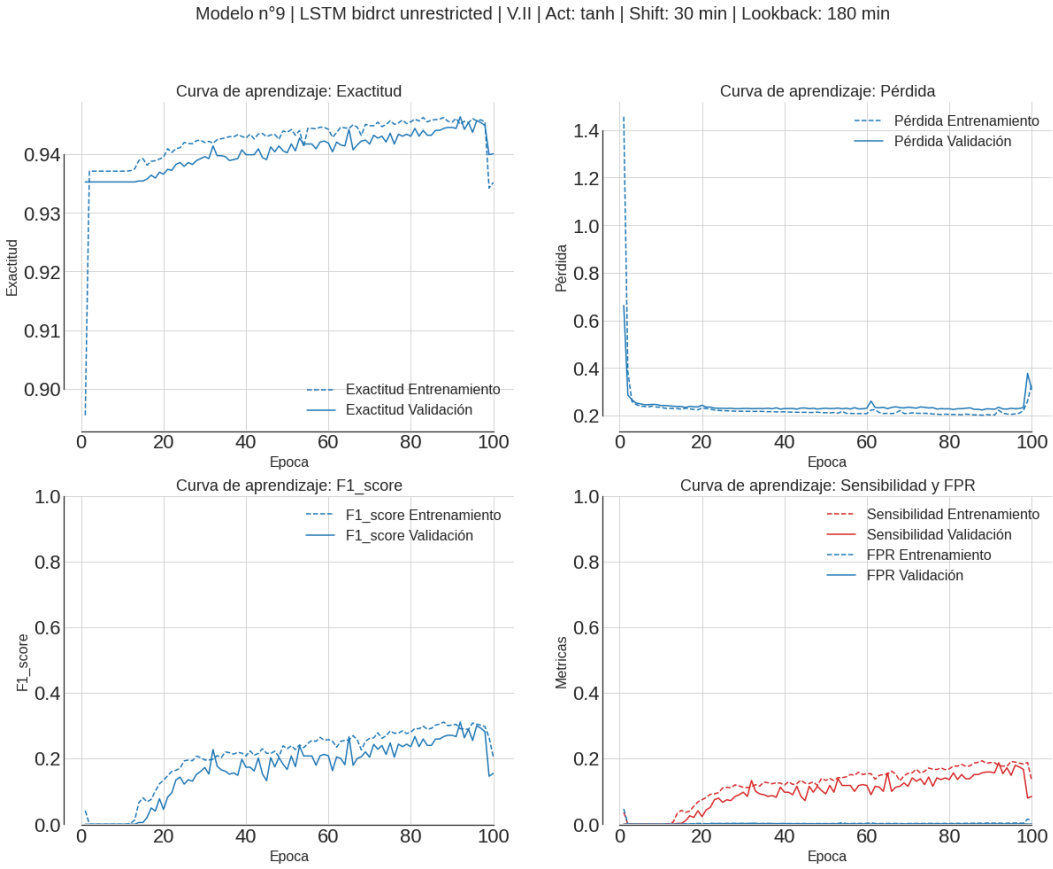


Figura 4.85: Modelo definitivo LSTM Versión II.

Las métricas de testeo (Tabla 4.38) también confirman el pobre rendimiento del modelo con una sensibilidad de apenas 0.16 y un F1-score de 0.28.

Tabla 4.38: Métricas en conjunto de testeo modelo definitivo LSTM Versión II.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.2935	0.9464	0.1646	0.2811	0.0004

Las señales temporales conocidas (Figura 4.87) y no conocidas (Figura 4.88) evidencian el bajo desempeño del modelo, que es incluso peor al desempeño de los modelos *MLP* que no consideran patrones temporales para generar las predicciones.

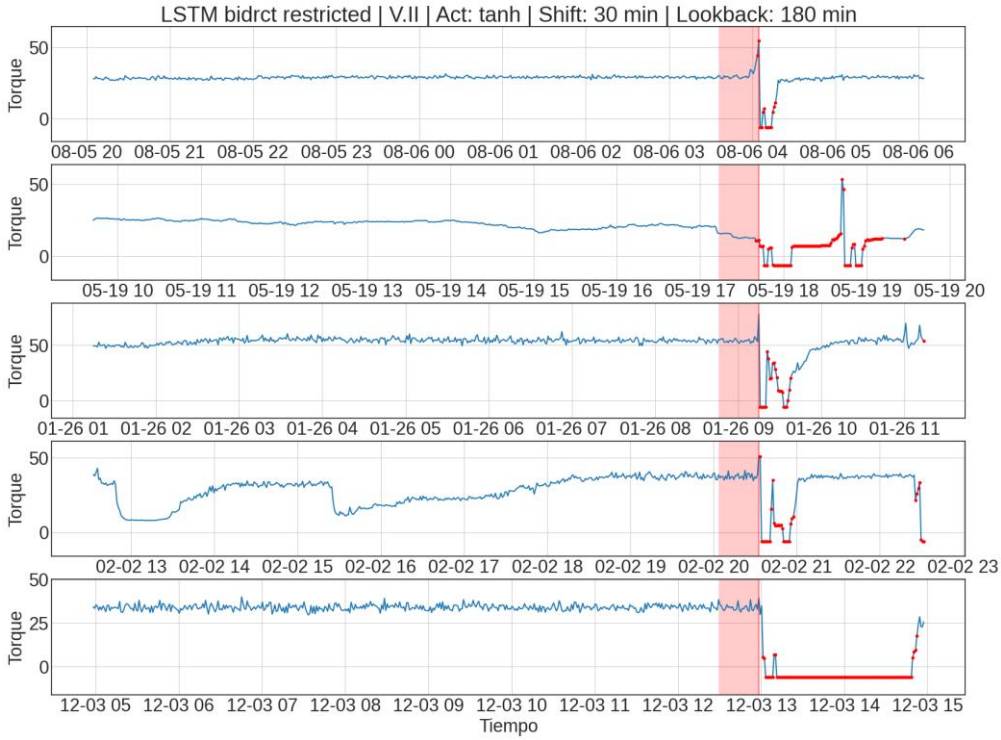


Figura 4.86: Señales temporales de algunos eventos evaluadas en modelo definitivo LSTM Versión II.

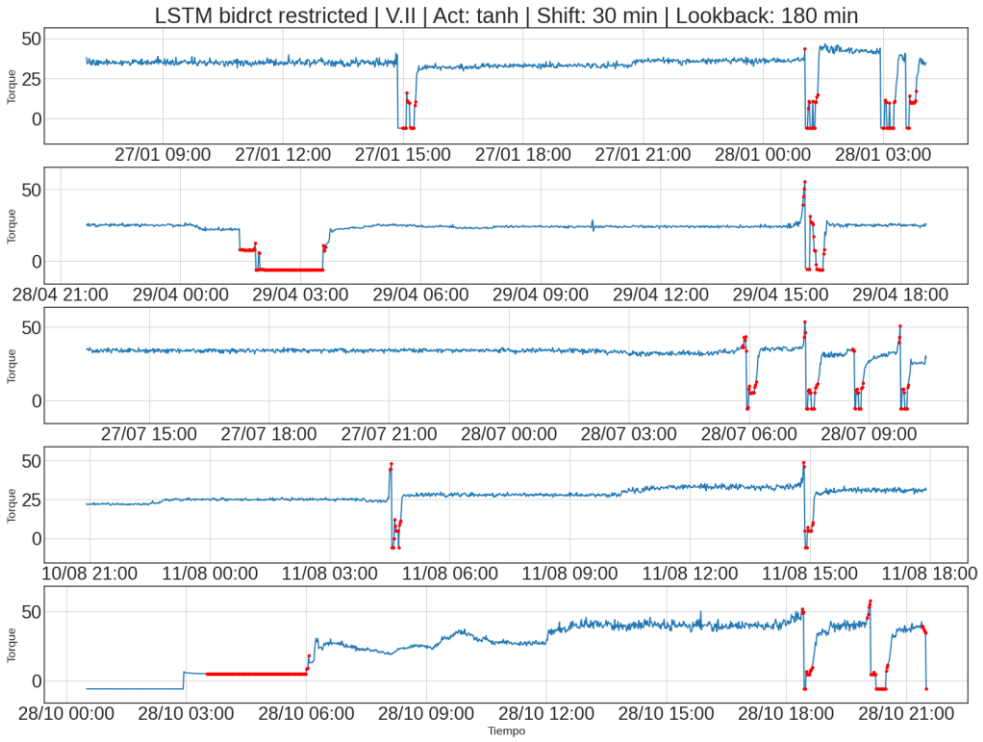


Figura 4.87: Señales temporales evaluadas en modelo definitivo LSTM Versión II.

Modelos con capas Convolucionales

De igual manera se intentó reducir el sobreajuste en los modelos convolucionales, implementando las medidas anteriormente aplicadas en los modelos recurrentes y añadiendo otras.

- Reducir número de capas y neuronas
- Reducir *batch*
- Agregar regularización L1 y *dropout*
- Reducir learning rate
- Cambiar el optimizador

Se entrenaron un total de 19 modelos implementando las medidas anteriores y con los siguientes parámetros.

- Función de activación: *relu*
- Muestras positivas (minutos previos al evento): 30 minutos
- Tamaño de ventana temporal: 240 minutos

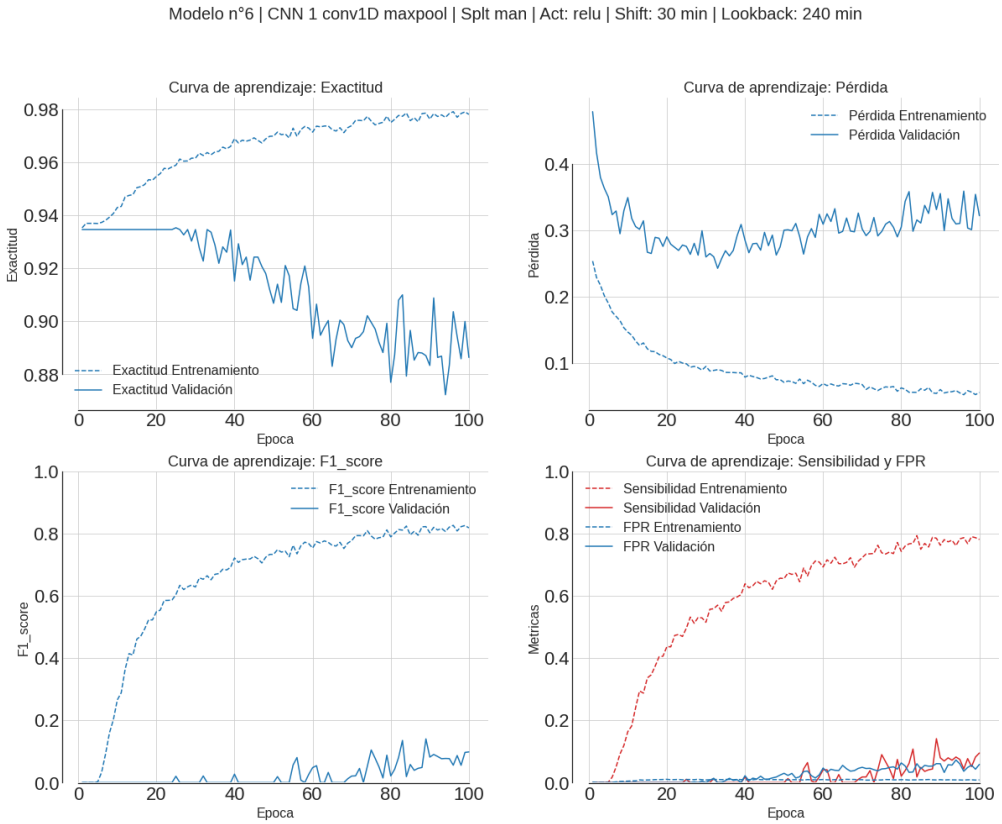


Figura 4.88: Modelo definitivo CNN Versión II.

Tabla 4.39: Métricas modelo definitivo CNN Versión II.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.4411	0.8746	0.1417	0.1267	0.0751

Entre los 19 modelos entrenados ninguno entregó un resultado satisfactorio en términos de curvas de aprendizaje. Los resultados mostrados en la Figura 4.89 corresponden al modelo que mejor rendimiento tuvo en comparación con el resto. Sin embargo, se observa que la pérdida de validación es creciente, indicando sobreajuste. Tanto la sensibilidad como el F1-score en el conjunto de validación son inferiores a 0.1, mientras que en el conjunto de entrenamiento estas métricas alcanzan valores en torno a 0.8. Los resultados de testeo (Tabla 4.39) reafirman el mal desempeño del modelo en datos no conocidos.

La evaluación del modelo con señales temporales conocidas (Figura 4.90) pareciera tener un correcto desempeño, sin embargo, con señales no conocidas (Figura 4.91) la capacidad predictiva del modelo pareciera ser nula.

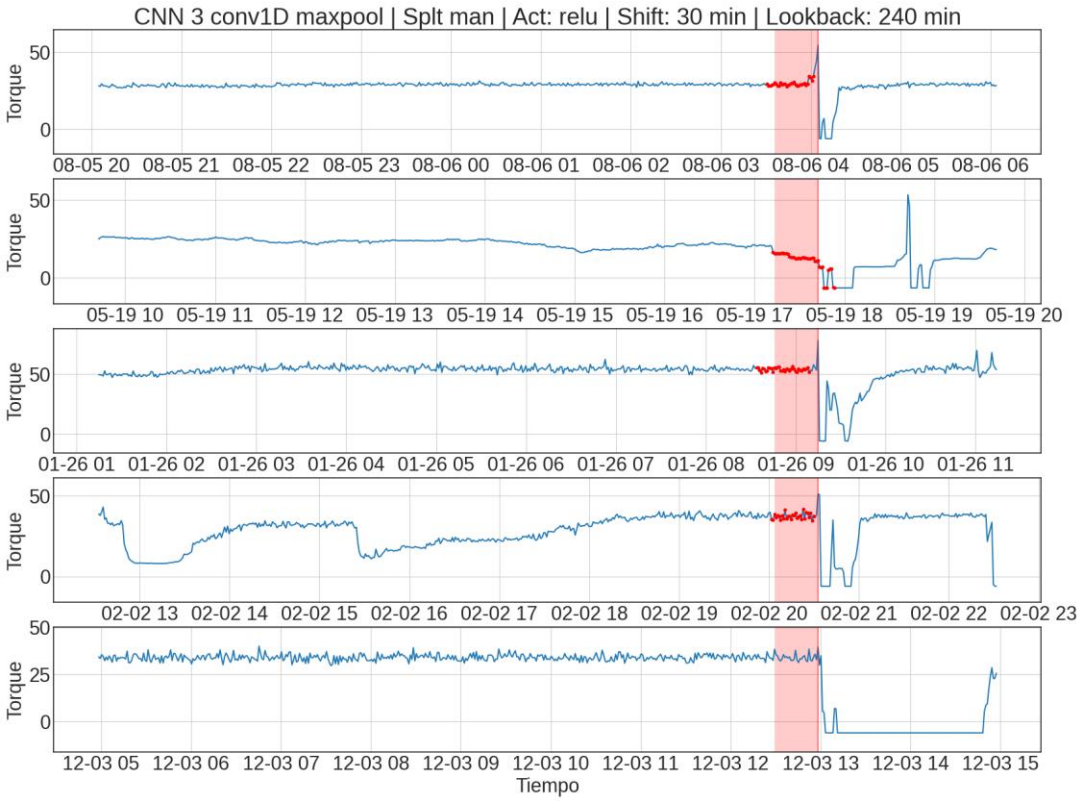


Figura 4.89: Señales temporales de algunos eventos evaluadas en modelo definitivo CNN Versión II.

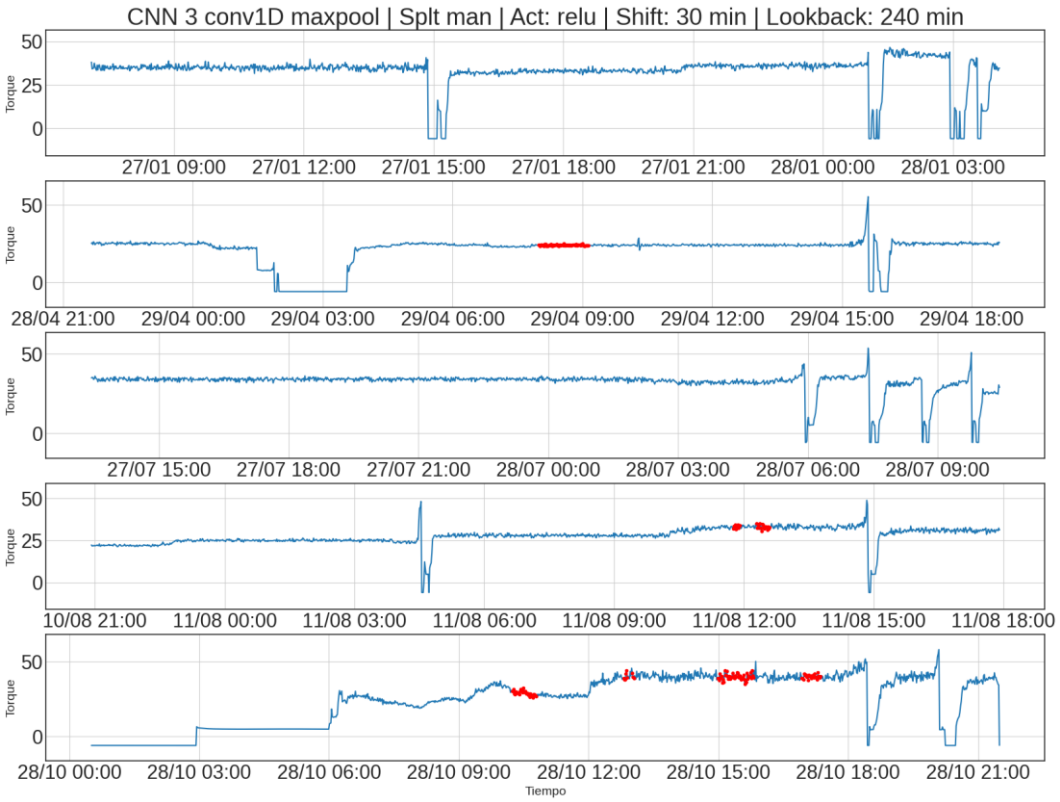


Figura 4.90: Señales temporales evaluadas en modelo definitivo CNN Versión II.

4.3.9. Modelos con capas recurrentes y convolucionales: Versión III

Una última versión se desarrolló para facilitar el entrenamiento de los modelos, realizando una limpieza de los datos de entrenamiento, esto con el fin de mejorar el rendimiento de los modelos, ya que por las vías anteriores no se llegó a buenos resultados.

Debido a que los datos se tratan como una señal única, la temporalización de los datos origina algunas ventanas temporales como la observada en la Figura 4.92, donde el evento de alto torque está al principio o en alguna parte del medio de la ventana. Este es un gran problema ya que la ventana *j* contiene un evento de alto torque (positivo), pero es etiquetada en base a la etiqueta del último elemento de la ventana (negativo). Lo anterior genera confusión en el entrenamiento de los modelos y no permite un aprendizaje eficiente.

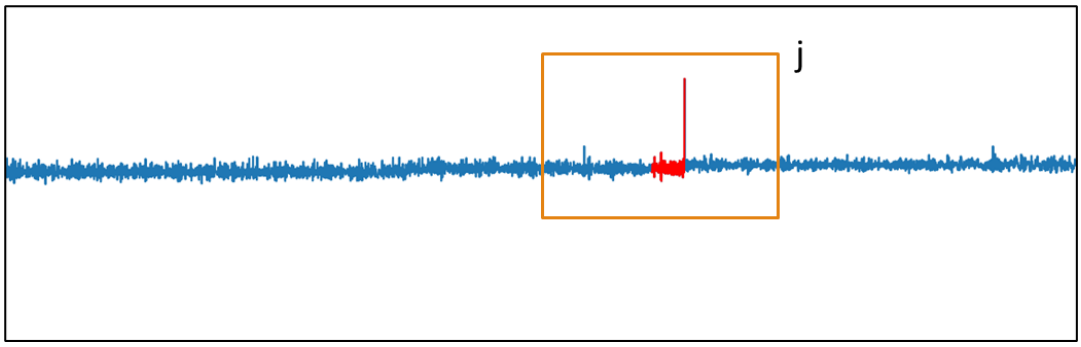


Figura 4.91: Ventana j es etiquetada negativa a pesar de contener un evento positivo.

Por los motivos anteriormente explicados, se modifica ligeramente la temporalización de los datos de manera que la extracción de ventanas temporales se hace para cada evento por separado, tal como se ejemplifica en la Figura 4.93.

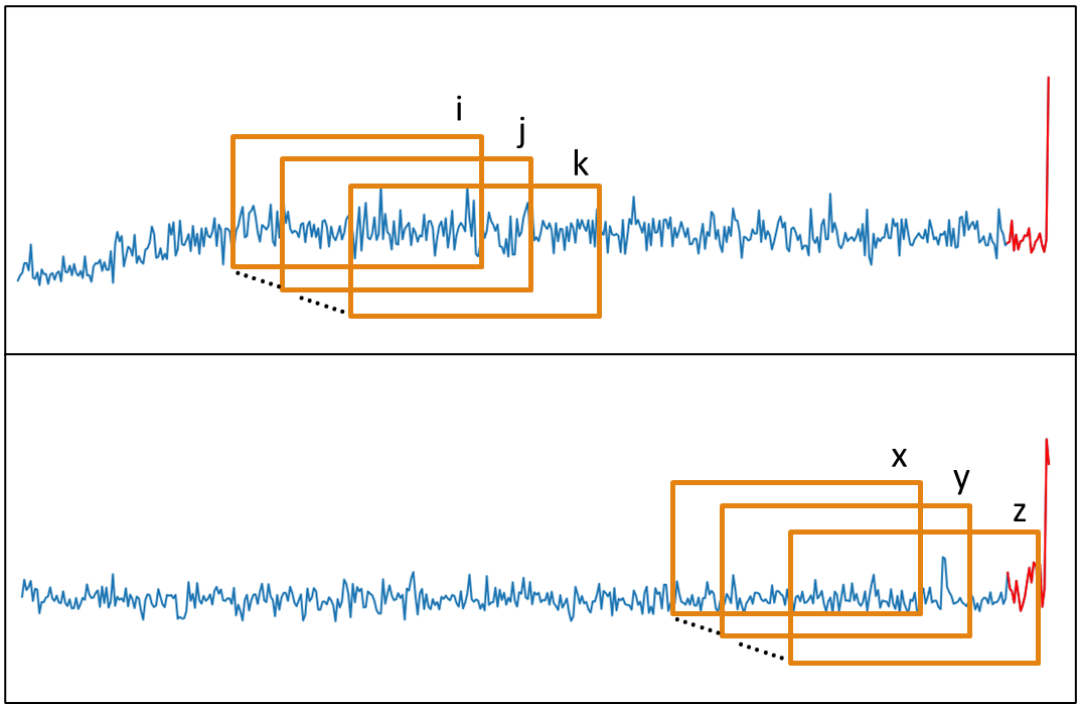


Figura 4.92:Esquema de las ventanas generadas por temporalización en la Versión III.

Considerando lo anterior, el procesamiento de los datos se ve de la siguiente forma.

1. Cada evento de alto torque se toma como un conjunto de datos independiente.

2. Etiquetado de los datos.
3. Disminuir frecuencia de muestreo.
4. Dividir los eventos en conjuntos de Entrenamiento, Validación y Testeo.
5. Temporalizar cada señal de cada conjunto de datos de forma independiente.
6. Normalizar datos a partir de los datos de Entrenamiento.

Se entrenaron 5 modelos con arquitecturas más simples (ver Anexo) para sobrellevar el sobreajuste observado en los casos estudiados, aplicando regularización en algunos modelos. Cabe destacar que, por la extensión de este trabajo, no se exploraron tantas arquitecturas y ajustes de parámetros con esta estrategia de preparación de datos, proponiéndose como trabajo futuro el ajustar de mejor forma la topología de las redes utilizando la metodología de procesamiento propuesta en esta última versión, la cual incorpora además la mejora fundamental mencionada al inicio de la sección 4.3.8.

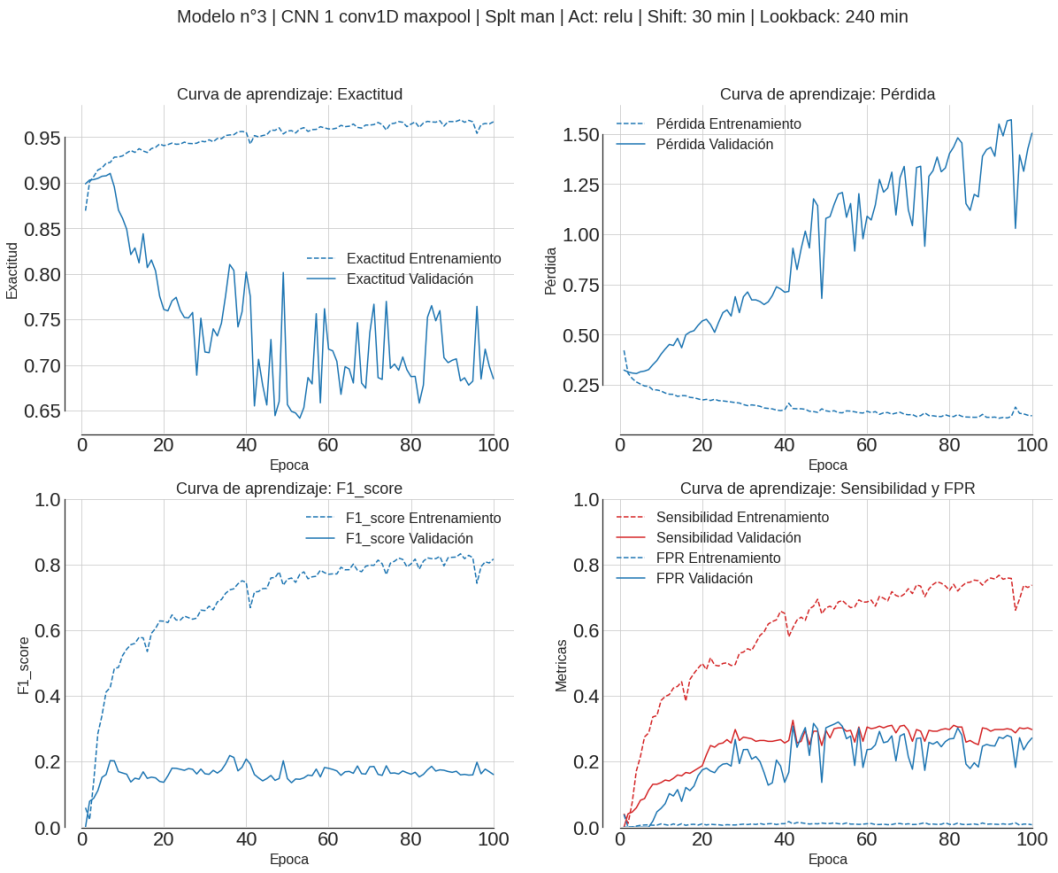


Figura 4.93: Curvas de aprendizaje modelo definitivo LSTM Versión III.

Las curvas de aprendizaje del modelo seleccionado se muestran en la Figura 4.94. La curva de pérdida de validación muestra un comportamiento creciente indicando sobreajuste. Esto se reafirma al observar las curvas de sensibilidad y F1-score, donde para el conjunto de entrenamiento sus valores son cercanos a 0.8 mientras que para el conjunto de validación los resultados son inferiores a 0.3.

Tabla 4.40: Métricas en conjunto testeo modelo definitivo LSTM Versión III.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.7668	0.8285	0.4625	0.3496	0.1309

Las métricas promedio calculadas en el conjunto de testeo son mostradas en la Tabla 4.40, donde la tendencia es similar a lo observado en el conjunto de validación.

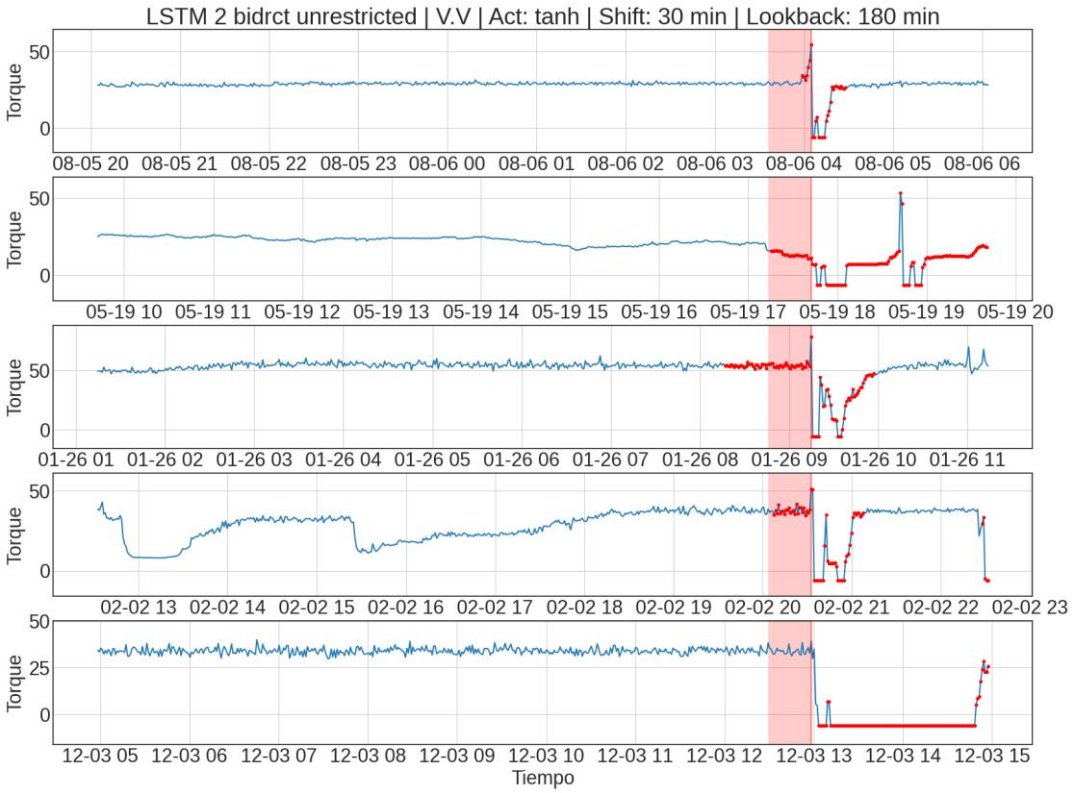


Figura 4.94: Señales temporales de algunos eventos evaluadas en modelo definitivo LSTM Versión III.

Las señales temporales conocidas evaluadas por el modelo (Figura 4.95) muestran algunos resultados correctos, mientras que en otros eventos el modelo no anticipa el alto torque. Por otro lado, al evaluar señales temporales no conocidas (Figura 4.96), en algunos casos logra identificar algunos eventos de alto torque, sin embargo, pareciera ser un resultado azaroso.

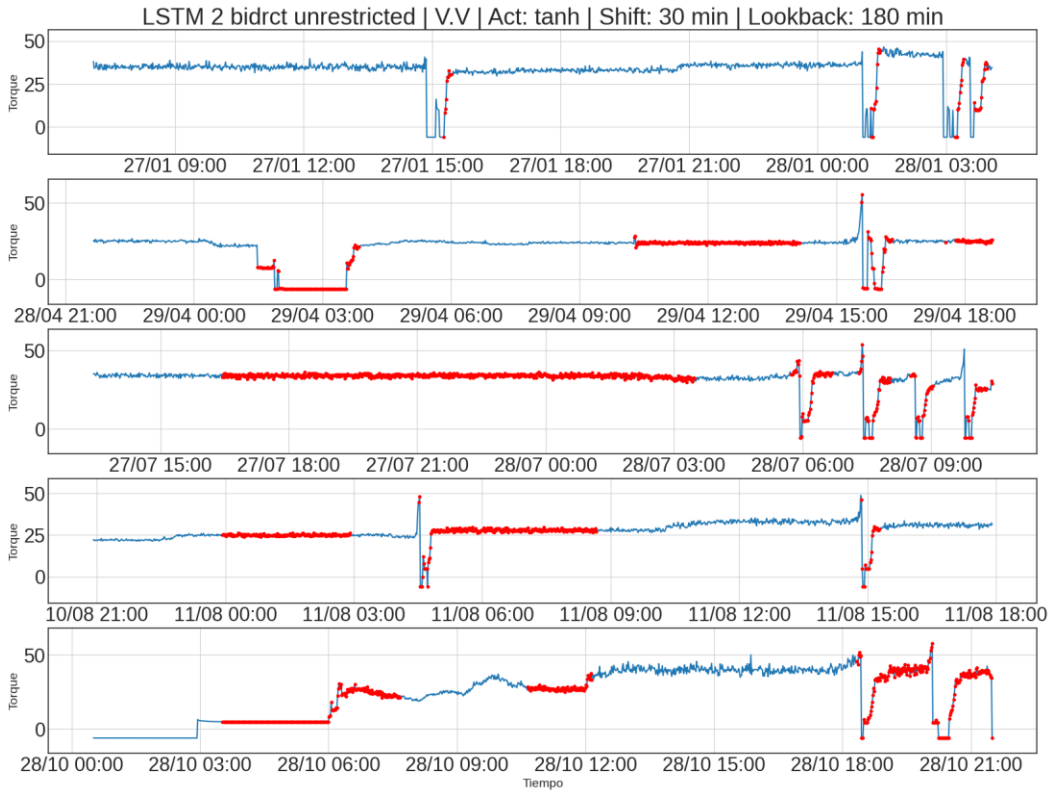


Figura 4.95: Señales temporales evaluadas en modelo definitivo LSTM Versión III.

Modelos convolucionales

En cuanto a los modelos convolucionales solamente se alcanzó a desarrollar un modelo cuya arquitectura es resumida en la Figura 4.97.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 221, 16)           5776

 dropout (Dropout)           (None, 221, 16)           0

 max_pooling1d (MaxPooling1D  (None, 22, 16)           0
 )

 flatten (Flatten)           (None, 352)               0

 dense (Dense)               (None, 32)                11296

 output (Dense)              (None, 1)                 33

=================================================================
Total params: 17,105
Trainable params: 17,105
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.96: Arquitectura modelo definitivo CNN Versión III.

Las curvas de aprendizaje mostradas en la Figura 4.98 muestran un modelo sobreajustado, con un excelente rendimiento en los datos de entrenamiento, pero con un desempeño todavía mejorable en los datos de validación, aun cuando el F1-score es el mejor logrado de entre los modelos evaluados en las Versiones I y II, y se logra estabilizar la pérdida del conjunto de validación (no creciente). Este último indicador permanece en torno a 0.35.

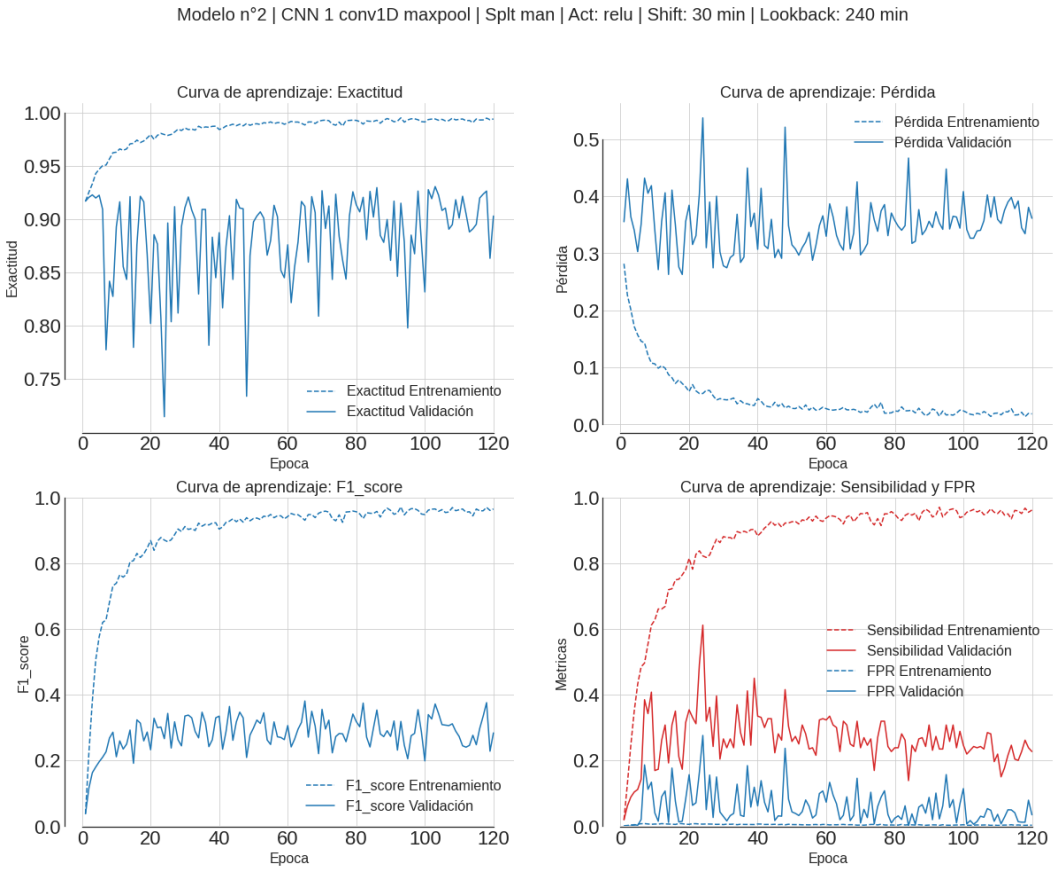


Figura 4.97: Modelo definitivo CNN Versión III.

Tabla 4.41: Métricas conjunto de testeo modelo definitivo CNN Versión III.

Pérdida	Exactitud	Sensibilidad	F1-score	FPR
0.4529	0.9194	0.2249	0.3165	0.0178

Las métricas promedio calculadas en el conjunto de testeo (Tabla 4.41) también evidencian un rendimiento bajo lo deseado al evaluar el modelo en datos no conocidos.

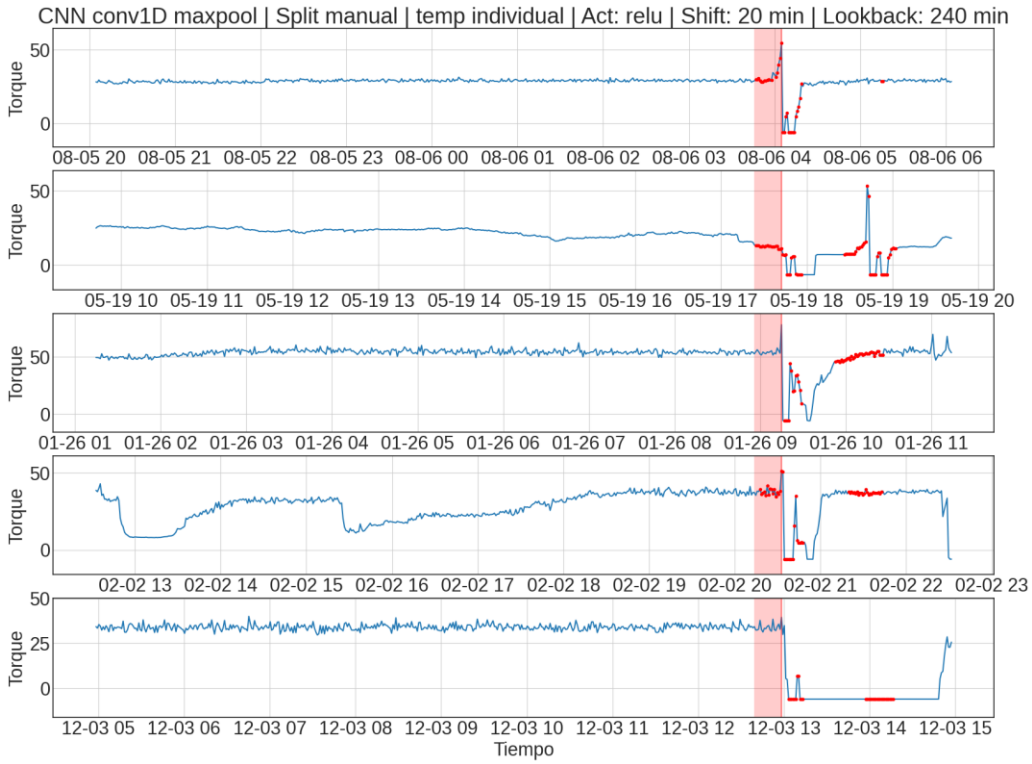


Figura 4.98: Señales temporales de algunos eventos evaluadas en modelo definitivo CNN Versión III.

En cuanto a los eventos de la Figura 4.99, los 3 primeros eventos son parte del conjunto de entrenamiento, el 4° evento pertenece al conjunto de validación, y último al conjunto de testeo.

4.3.10.Trabajo futuro

Con la finalidad de mejorar el rendimiento de los modelos desarrollados para el monitoreo de las prensas de desplazamiento se propone lo siguiente como trabajo futuro.

- Mejorar la calidad de los datos de entrada mediante la agrupación de eventos de alto torque con características similares, previo al entrenamiento de los modelos.
- Utilizar métodos de selección y extracción de parámetros como *Feature Engineering*, *forward/backward selection*, y ampliar el alcance de las señales utilizadas en el monitoreo de una prensa, incluyendo variables más distantes aguas arriba o incluso aguas abajo en el proceso.
- Explorar nuevas arquitecturas manteniendo la metodología de procesamiento de datos planteada en las versiones II y III.
- Explorar nuevas estrategias de procesamiento de datos.
- Utilizar ventanas de tiempo más acotadas en la temporalización.
- Explorar y evaluar el desempeño de otros tipos de modelos de aprendizaje de máquinas como *Autoencoders*.

5. Conclusiones

Conclusiones generales

- El uso de modelos de aprendizaje de máquinas requiere una base de información que debe ser recolectada de forma planificada y con la debida anticipación. Este trabajo aprovechó aquella información que ya había sido registrada por períodos de meses y años en Planta Nueva Aldea para un grupo de equipos, pero se advierten brechas que limitan la aplicación de modelos a otros equipos que no están siendo sistemáticamente monitoreados. Por tanto, el avance hacia una industria y un mantenimiento 4.0 debe hacerse de forma planificada y con la debida anticipación para ser exitosa.

- Es muy importante asegurar la calidad de los datos para el desarrollo de modelos de aprendizaje de máquinas que sean robustos. Algunas consideraciones a tener en cuenta son:

    - Registrar eventos y fallas de forma adecuada para evitar trabajos de corrección al momento de utilizar estos registros para sus respectivos análisis, ya sea por errores de tipeo o mala clasificación de los datos.
    - Asegurar que las variables operacionales monitoreadas contengan mediciones correctas, ya que se dio el caso de que algunas variables parecían estar historizadas pero en la realidad no contenían datos reales.

- En este trabajo se desarrollaron y evaluaron modelos de aprendizaje de máquinas *offline*. La implementación *online* requiere la integración con alguna plataforma informática adecuada donde sea posible ejecutar estos modelos de forma continua y que permita leer y escribir información desde y hacia las bases de datos actualmente implementadas en Arauco.

- La realización de este trabajo de investigación en conjunto con la empresa Arauco representó un enorme desafío personal donde se aplican los conocimientos teóricos adquiridos durante la formación académica a problemas reales presentes en la industria, entregando una visión más integral y ayudando en la transición desde el mundo académico universitario al mundo laboral.

- Entre los aprendizajes logrados durante este trabajo se destaca:

- La importancia de planificar un trabajo de forma detallada antes de realizarlo y plantear objetivos claros, ya que si esto no se especifica desde un principio es fácil desviarse durante el desarrollo del trabajo.
- La importancia de asesorarse con expertos o gente de más experiencia cuando existen dudas o no se tiene claridad en lo que se está realizando.
- La importancia de llevar un orden y un registro del trabajo que se va realizando, para evitar realizar el mismo trabajo nuevamente, así como también tener clara la forma en que se obtuvieron determinados resultados y así poder replicarlos en el futuro.
- La importancia de tener experiencia en el campo de la Inteligencia Artificial y el *Machine Learning* por un lado en la selección de los modelos o algoritmos más adecuados según la aplicación que se quiera desarrollar, y por otro lado en el ajuste de parámetros y la interpretación de resultados.

Caso de estudio N°1: Turbomáquinas

Con respecto al caso de estudio: Turbomáquinas, se concluye lo siguiente:

- En principio, los modelos de detección desarrollados sí serían capaces de alertar la presencia de anomalías, como ineficiencias o sobrecargas en los equipos, cumpliéndose los objetivos específicos para esta familia de activos.

- En cuanto a los modelos de detección de novedades desarrollados, cuando el equipo opera fuera de la región normal definida por el modelo, se requiere un análisis más exhaustivo antes de apresurarse y concluir que se trata de una falla. No obstante, los modelos desarrollados son una buena herramienta para apoyar el monitoreo de la condición de un gran volumen de equipos, permitiendo identificar y acotar el número de equipos que requieren un análisis más avanzado de su condición por un humano, tanto en términos de mantenimiento como de operación.

- En cuanto a los algoritmos de agrupamiento, las métricas no supervisadas: coeficiente de Silueta e Índice Calinski-Harabaz, no son un buen indicador del desempeño de algoritmos de agrupamiento basados en densidad como DBSCAN.

- Los algoritmos de detección de anomalías Elliptic Envelope y Local Outlier Factor son los que mejor desempeño tienen en los datos de operación de las turbomáquinas estudiadas, mientras que el algoritmo Isolation Forest si bien tuvo un rendimiento relativamente bueno en algunos casos particulares, el desempeño general es bajo y no es un buen modelo para los casos de aplicación.

- El ajuste de parámetros en el algoritmo OC-SVM es más complicado en comparación con los otros algoritmos. No fue posible obtener buenos resultados para los casos de aplicación estudiados y por lo tanto no se recomienda su uso para el monitoreo generalizado de un gran volumen de equipos.

- La frontera de decisión del algoritmo Elliptic Envelope define una región normal continua pero muy rígida (elipse) lo que no permite un buen ajuste a las curvas asociadas a la operación normal de las turbomáquinas.

- La frontera de decisión del algoritmo Local Outlier Factor define una región que se ajusta muy bien a los datos normales, pero con el inconveniente de generar discontinuidades en la región. El uso de este algoritmo podría generar falsas alarmas referentes a la condición de un equipo.

- Las métricas supervisadas no aportan demasiado en cuanto al desempeño real de cada modelo de detección de novedades, al menos en estos casos de aplicación particular, donde el objetivo es encontrar una región de operación bien ajustada a los datos normales y sin discontinuidades, por tanto, la herramienta de evaluación más valorada es la visualización de la frontera de decisión.

- Para el desarrollo de modelos de detección de novedades, es muy importante que los datos de entrenamiento contengan la mayor información posible sobre la operación normal del equipo, es decir, datos en todo el rango de operación normal, de modo que la región de operación normal definida por el modelo sea representativa evitando la generación de falas alarmas cuando el equipo opere en nuevos puntos que son realmente normales, pero no han sido registrados en el pasado.

- Es muy importante tener conocimiento de los datos que realmente corresponden al comportamiento normal de un equipo y sus modos de operación. En los casos estudiados se asumió que grupos de datos (*clusters*) más poblados corresponden a datos asociados a una operación normal, lo que se fundamenta en que efectivamente en Planta Nueva Aldea los equipos son requeridos con una alta disponibilidad para mantener la continuidad del proceso productivo.

- Los modelos de detección de novedades desarrollados funcionan mejor en equipos de velocidad variable que son operados mediante un lazo de control automático, ya que los datos se mueven en un rango continuo, mientras que con equipos operados de forma manual los datos no se agrupan en una región continua, lo que hace difícil determinar cuáles grupos de datos corresponden a la operación normal.

Caso de estudio N°2: Prensas de desplazamiento

Con respecto al caso de estudio: Prensas de desplazamiento, se concluye lo siguiente.

- El objetivo específico del desarrollo de modelos para detección temprana de eventos raros fue logrado con algunas salvedades. Los modelos desarrollados son capaces de detectar eventos de alto torque con hasta 1 hora de anticipación cuando se trata de eventos conocidos, es decir, son parte del conjunto de entrenamiento. Sin embargo, a pesar del uso de métodos tradicionales para limitar el sobreajuste, no fue posible mejorar la capacidad de generalización de los modelos, limitando su capacidad predictiva en el monitoreo continuo de las prensas de desplazamiento pudiendo solamente identificar anomalías al momento en que estas ocurren.

a) Modelos base: *Multi-Layer Perceptron*

- Los modelos base desarrollados corresponden a redes neuronales multicapa *MLP* que no consideran el orden temporal de los datos y por lo tanto no son capaces de predecir los eventos de alto torque con anticipación, solamente pueden identificar los peaks de torque en el momento que ocurren y por lo tanto no resultan de utilidad para tomar acción anticipada y evitar dichos eventos.

- Los métodos de balance de clases mejoran el rendimiento de los modelos en términos de la métrica sensibilidad, pero a costa de incrementar la presencia de falsos positivos.

- Utilizar regularización por *dropout* puede ser contraproducente. Cuando se usan valores altos (0.5) el rendimiento de los modelos por lo general disminuye, de manera que se recomienda usar valores más moderados (0.2) cuando se quiere reducir el sobreajuste de un modelo.

- El mejor modelo *MLP* que se desarrolló alcanza una sensibilidad del 58%, un F1-score de 0.7, una exactitud de aproximadamente 99% y con una tasa de falsos positivos de apenas 0.002.

b) Modelos con capas recurrentes y convolucionales: Versión I

- Uno de los hallazgos más relevantes encontrados en este caso de estudio es que realizar la temporalización de los datos antes de separarlos en los conjuntos de entrenamiento, validación y testeo, es el principal causante del sobreajuste de los modelos y su baja capacidad de generalización, debido a que los 3 conjuntos de datos contienen información muy similar.

- El mejor modelo recurrente desarrollado en la Versión I corresponde a la variante *LSTM Bidirectional*, y alcanza una sensibilidad del 98%, un F1-score de 0.94, una exactitud cercana al 99% y una tasa de falsos positivos de 0.014.

- El mejor modelo convolucional desarrollado en la Versión I alcanza una sensibilidad del 97%, un F1-score de 0.96, una exactitud del 99% y una tasa de falsos positivos de 0.006.

- Las métricas y curvas de aprendizaje por sí solas no son un buen indicador de que el modelo presenta un buen desempeño en la predicción o detección temprana de los eventos de alto torque. Mientras que la evaluación de los modelos en señales del dominio del tiempo fue la herramienta de evaluación que permitió encontrar el sobreajuste.

c) Modelos con capas recurrentes y convolucionales: Versión II

- Los modelos recurrentes y convolucionales de la Versión II fueron desarrollados para solucionar el problema de sobreajuste descubierto en la Versión I, mediante la separación de los datos en los conjuntos de entrenamiento, validación y testeo antes de realizar la temporalización de estos, evitando así generar conjuntos con información similar.

- Los métodos de regularización L1 y L2 en conjunto con la implementación de arquitecturas más simples (menos capas y menos neuronas), son las herramientas más efectivas para combatir el sobreajuste, aunque esto se consigue a costa de reducir el desempeño del modelo en términos de pérdida y también al evaluar el modelo en el dominio del tiempo.

- El mejor modelo recurrente desarrollado en la Versión II alcanza una sensibilidad del 16%, con un F1-score de 0.28, una exactitud del 95% y una tasa de falsos positivos de 0.0004. El mejor modelo convolucional desarrollado en la Versión II solamente alcanza una sensibilidad del 14%, un F1-score de 0.13, una exactitud del 87% y una tasa de falsos positivos de 0.08.

d) Modelos con capas recurrentes y convolucionales: Versión III

- Los modelos recurrentes y convolucionales de la Versión III fueron desarrollados con la finalidad de hacer más eficiente el entrenamiento de los modelos, mediante la modificación de la temporalización de los datos. Sin embargo, esta versión no fue muy explorada dada la extensión de este trabajo, por tanto, los resultados obtenidos muestran que aún existe potencial para alcanzar un mejor desempeño siguiendo esta estrategia de procesamiento de datos.

- El mejor modelo recurrente desarrollado en la Versión III alcanza una sensibilidad del 46%, con un F1-score de 0.35, una exactitud del 83% y una tasa de falsos positivos de 0.13, mientras que el único modelo convolucional desarrollado en la Versión III alcanza una sensibilidad del 22%, un F1-score de 0.32, una exactitud del 92% y una tasa de falsos positivos de 0.018.

Bibliografía

[1] Superintendencia de Procesos [2022]. Celulosa Arauco y Constitución S.A., Planta Nueva Aldea. [Presentación].

[2] Mataix, C. 1986. MECANICA DE FLUIDOS Y MAQUINAS HIDRAULICAS. Segunda edición. Madrid.

[3] JReyes Consultores Ltda. [2005] PLANTA ITATA HORNO DE CAL (554). [Manual]

[4] JReyes Consultores Ltda. [2005] PLANTA ITATA LAVADO Y DESLIGNIFICACION (546). [Manual]

[5] N. Véjar, Marcelo Viera [2005]. MANTENCIÓN MECÁNICA PRENSAS DE DESPLAZAMIENTO. [Manual]

[6] JReyes Consultores Ltda. [2005] PLANTA ITATA BLANQUEO (547). [Manual]

[7] Bassi, L. (2017). Industry 4.0: Hope, hype or revolution? 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI). doi:10.1109/rtsi.2017.8065927

[8] FACTORÍA ELECTRÓNICA. Tecnologías habilitadoras de la industria 4.0. Javier López [en línea] https://www.factoriadelfuturo.com/tecnologias-habilitadoras

[9] Artificial Intelligence and its Application in Different Areas Avneet Pannu, M. Tech Student Department of Computer Science Engineering DAV Institute of Engineering and Technology, Jalandhar India

[10] Chollet, F. 2018. Deep Learning with Python.

[11] S. Raschka y V. Mirjalili (2019). Aprendizaje automático con Python. Segunda edición en español.

[12] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one, 11(4):e0152173,2016

[13] Scikit-learn. Clustering. https://scikit-learn.org/stable/modules/clustering.html

[14] Sander J., Ester M, Kriegel H.P., Xu X. (1998) Density-Based Clustering in Spatial Databases: The Algortihm GDBSCAN and Its Applications.

[15] Rahmah, N., & Sitanggang, I. S. (2016). Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra. IOP

Conference Series: Earth and Environmental Science, 31, 012012. doi:10.1088/1755-1315/31/1/012012

[16] Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. Lecture Notes in Computer Science, 160–172. doi:10.1007/978-3-642-37456-2_14

[17] hdbscan. https://hdbscan.readthedocs.io/en/latest/index.html

[18] github. hdbscan. https://github.com/scikit-learn-contrib/hdbscan/blob/master/notebooks/Flat%20clustering.ipynb

[19] Scikit-Learn. Novelty and outlier detection. https://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection

[20] https://www.analyticsvidhya.com/blog/2022/06/one-class-classification-using-support-vector-machines/

[21] Basheer, I., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods, 43(1), 3–31. doi:10.1016/s0167-7012(00)00201-3

[22] Ranjan C. (2020) Understanding Deep Learning Application in Rare Event Prediction. First Edition.

[23] **https://www.r-bloggers.com/2020/12/weighting-confusion-matrices-by-outcomes-and-observations/**

[24] Scikit-Learn. Metrics. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[25] Scikit-Learn. Metrics. https://scikit-learn.org/stable/modules/clustering.html#calinski-harabasz-index

[26] UNE-EN 13306:2002: "Terminología del mantenimiento", versión oficial en español de la Norma Europea EN 13306, elaborada por el comité técnico AEN/CTN 151.

[27] T. Tinga, "Principles of Loads and Failure Mechanisms: Applications in Maintenance Realiability and Design", 2013, Capítulo 5

[28] L. Biggio and I. Kastanis (2020) Prognostics and Health Management of Industrial Assets: Current Progress and Road Ahead. doi: 10.3389/frai.2020.578613

[29] W. Olarte C., M. Botero A. y B. Cañon A. (2010) TÉCNICAS DE MANTENIMIENTO PREDICTIVO UTILIZADAS EN LA INDUSTRIA. Universidad Tecnológica de Pereira.

[30] M. Manfré 2020. Creation of a Machine Learning model for the Predictive Maintenance of an engine equipped with a rotating shaft. Electronic Engineering Master Degree Thesis. Politecnico Di Torino. 17-19 p.

[31] Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., and Siegel, D. (2014). Prognostics and health management design for rotary machinery systems-Reviews, methodology and applications. Mech. Syst. Signal Process. 42, 314–334. doi:10.1016/j.ymssp.2013.06.004

[32] Kadry, S. (2012). Diagnostics and prognostics of engineering systems: methods and techniques: methods and techniques. Hershey, PA: IGI Global.Google Scholar

[33] https://github.com/scikit-learn-contrib/hdbscan/blob/master/hdbscan/prediction.py

Anexo A

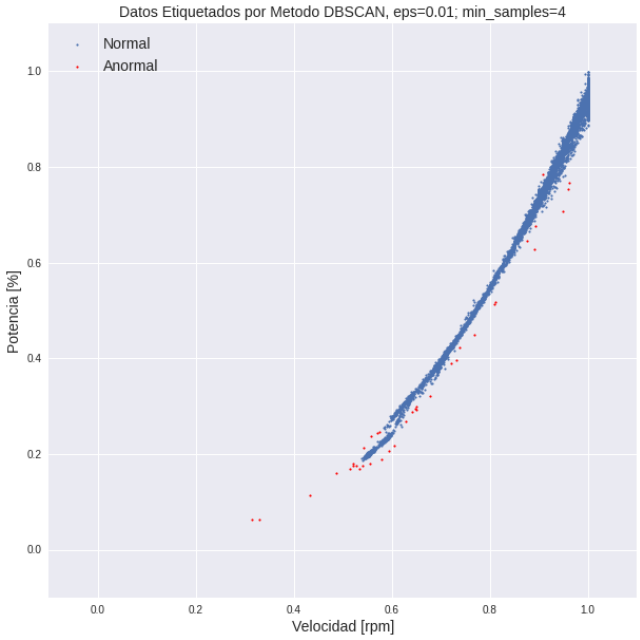
Tabla 0.1: Resumen de turbomáquinas estudiadas.

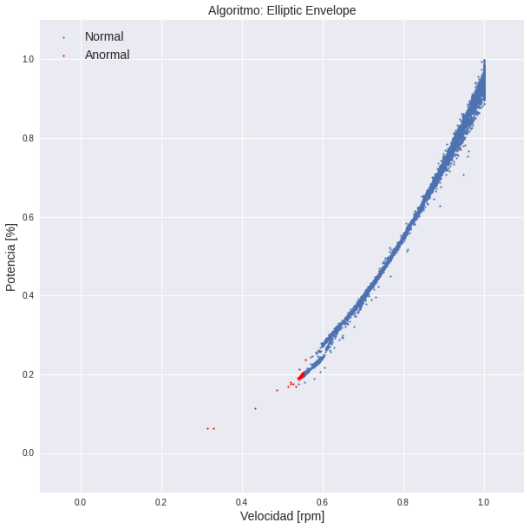
Turbomáquina	Tag del Equipo	Área o Sistema	Descripción
Bomba	552-21-303	Caldera Recuperadora	Bombas de transferencia de velocidad variable, impulsan el licor verde hacia el área de Caustificación
	552-21-302		
	572-21-1116	Máquina L1	Bomba baja consistencia, extrae la pulpa del estanque mezclador y la envía a harneros primarios
	572-21-2116	Maquina L2	
	572-21-1134	Máquina L1	Bomba fan, mezcla la pulpa con agua blanca para entregar un flujo continuo a presión y consistencia requeridas
	572-21-2134	Máquina L2	
	547-21-1202	Fibra L1	Bomba de pasta de media consistencia, de descarga de stand pipe prensa D0
	547-21-2202	Fibra L2	
	546-21-1211	Fibra L1	Bomba de pulpa de media consistencia, de descarga del reactor deslignificación con O2
	546-21-2211	Fibra L2	
Extractor	572-25-1464	Máquina L1	Ventiladores de extracción de aire del secador
	572-25-1465		

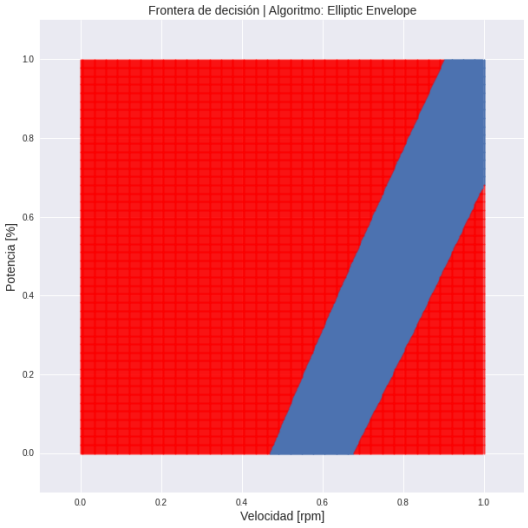
Turbomáquina	Tag del Equipo	Área o Sistema	Descripción
	572-25-2464	Máquina L2	
	572-25-2465		
Inductor	1461	Máquina L1	Ventiladores de inducción de aire al secador
	1462		
	2461	Máquina L2	
	2462		
Soplador	1355	Máquina L1	Sopladores de manta de la tercera prensa
	2355	Máquina L2	
Ventilador Tiro Inducido	554-25-152	Horno de Cal	Ventilador de evacuación de gases de combustión del Horno de Cal
	552-25-256	Caldera Recuperadora	Ventilador de extracción de gases de combustión desde hogar de Caldera Recuperadora
	552-25-257		
	552-25-258		

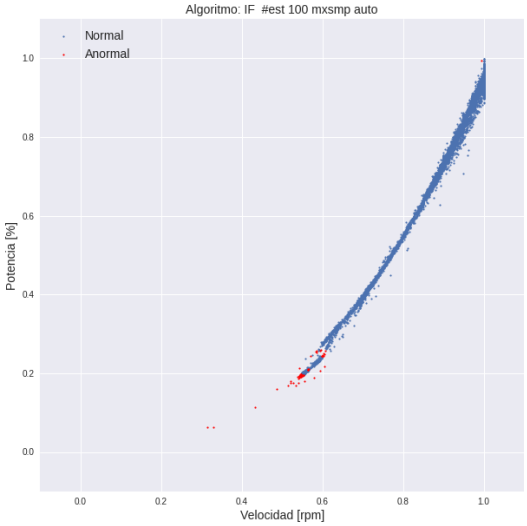
Etapa I

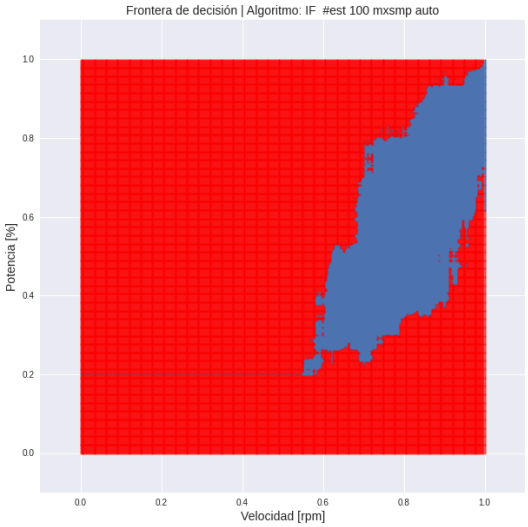
Bomba 302

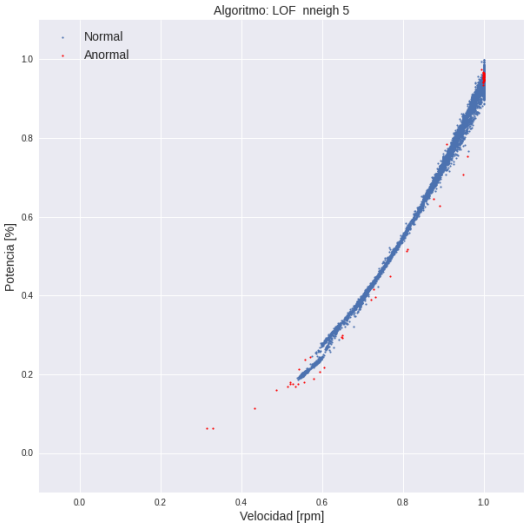


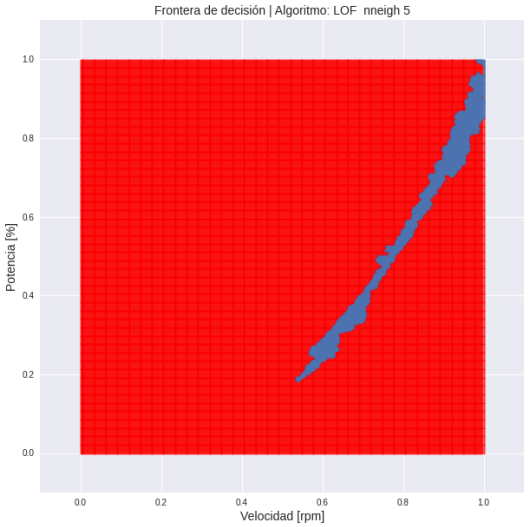


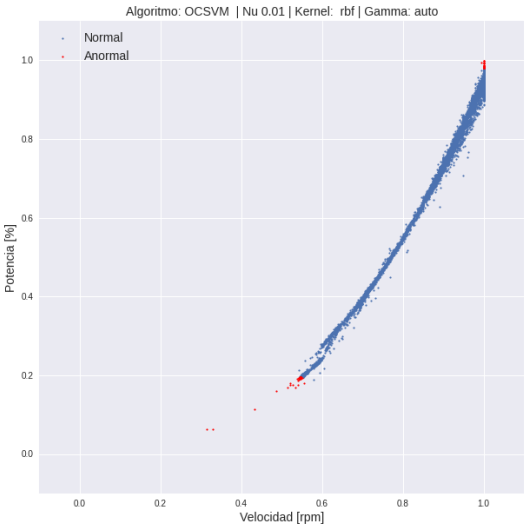


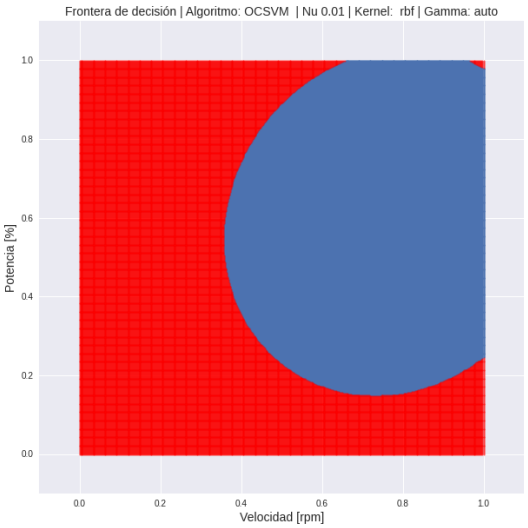

Algoritmo: IF #est 100 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 100 mxsmp auto
Potencia [%]
Velocidad [rpm]

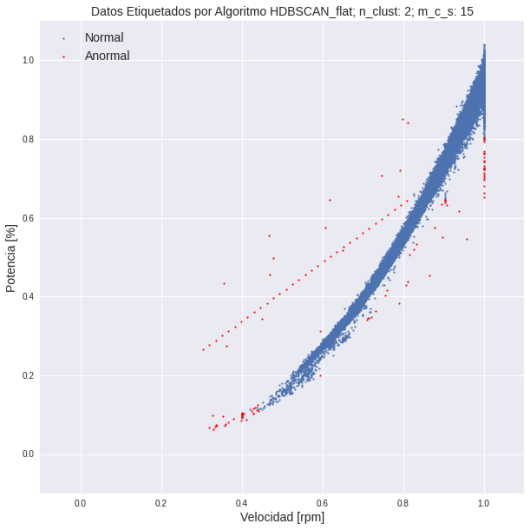

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

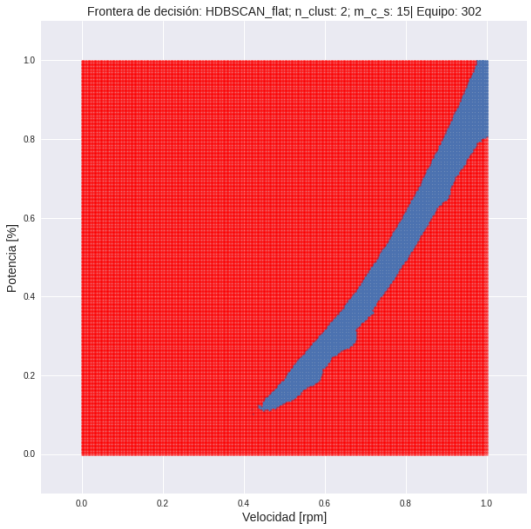

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

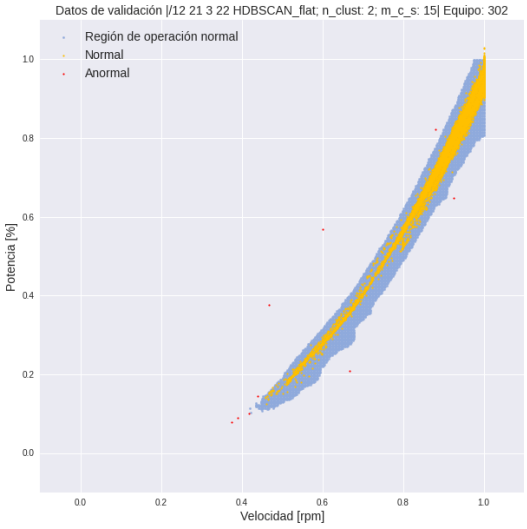

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

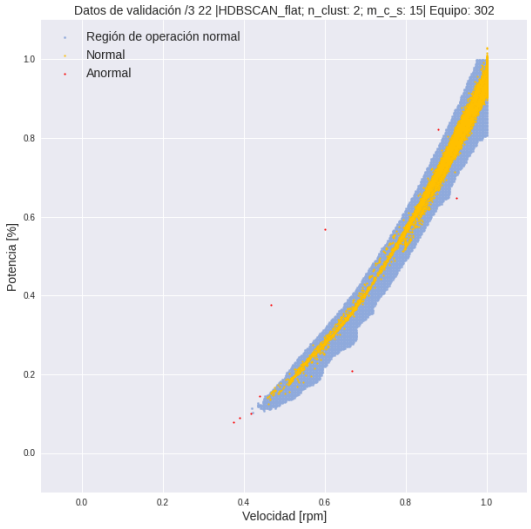

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9860	0.1410	0.2973	0.1913
Isolation Forest	0.9868	0.1928	0.4324	0.2667
Local Outlier Factor	0.9926	0.4167	0.8108	0.5505
OC-SVM	0.9862	0.1429	0.2973	0.1929

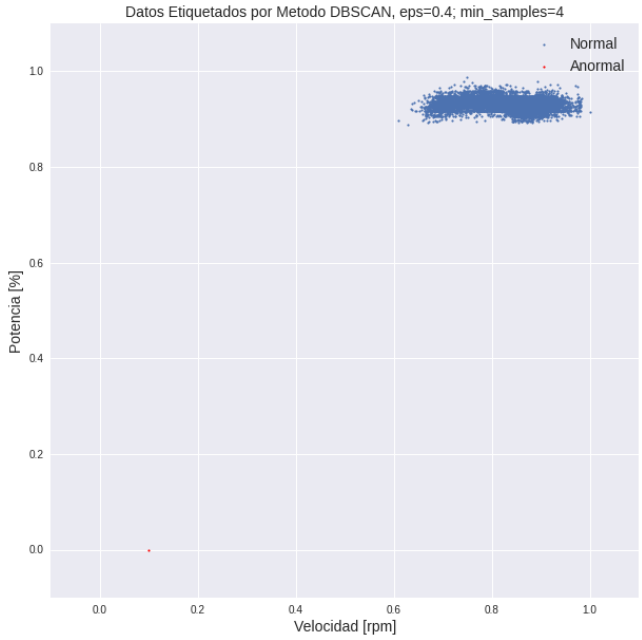


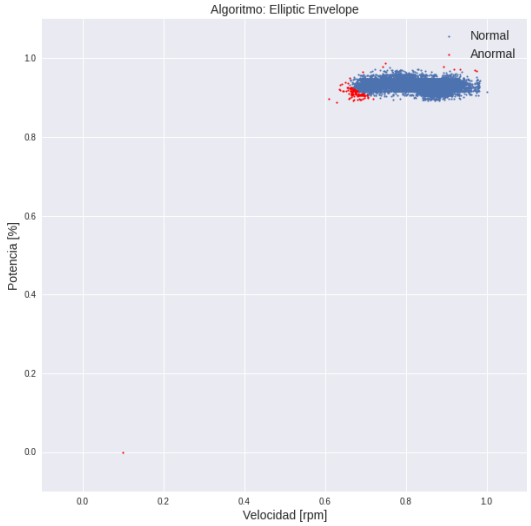


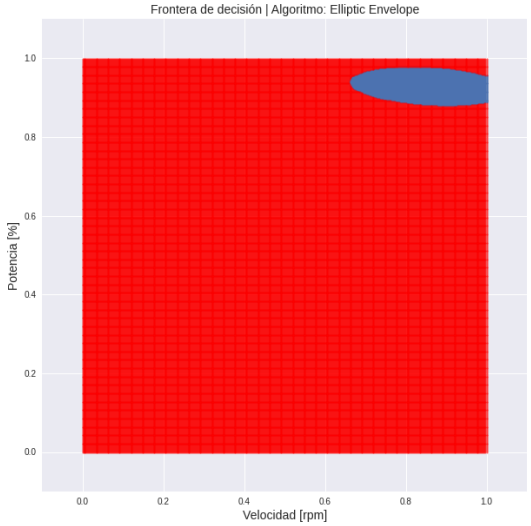


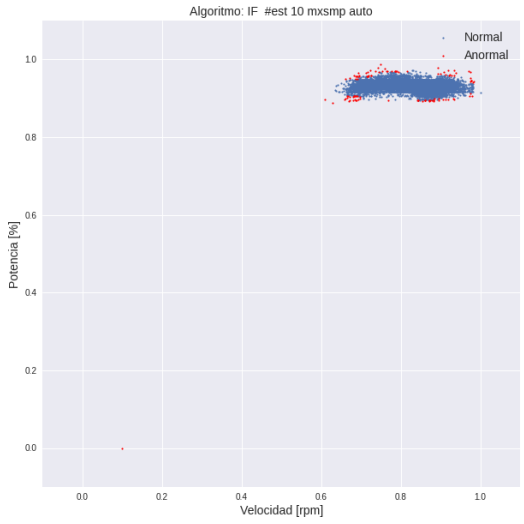


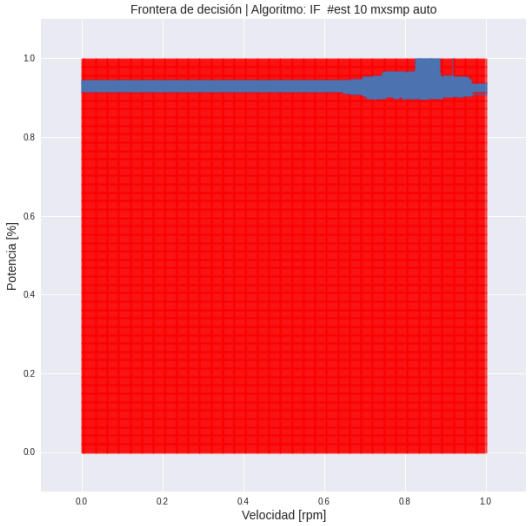
Bomba 1116

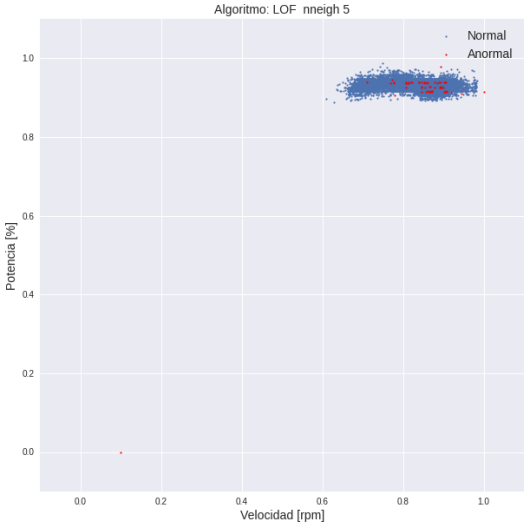

Datos Etiquetados por Metodo DBSCAN, eps=0.4; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]

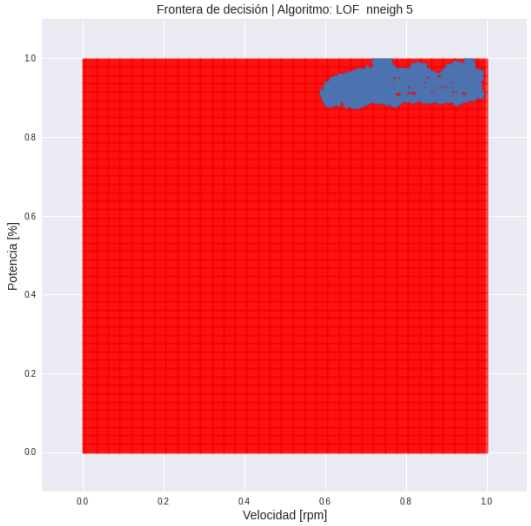

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]

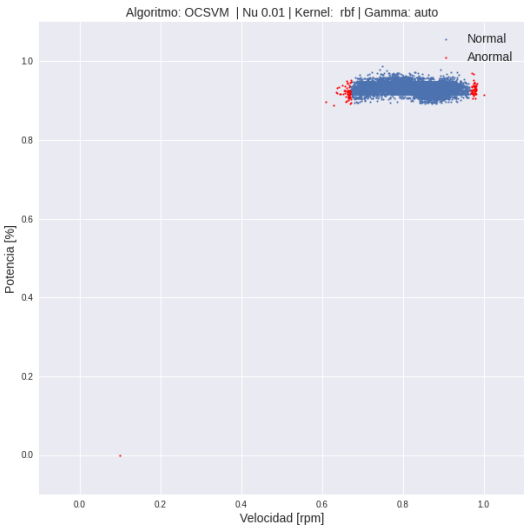

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]

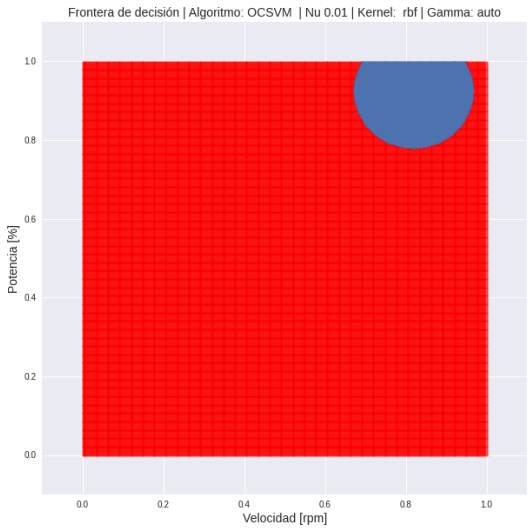

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

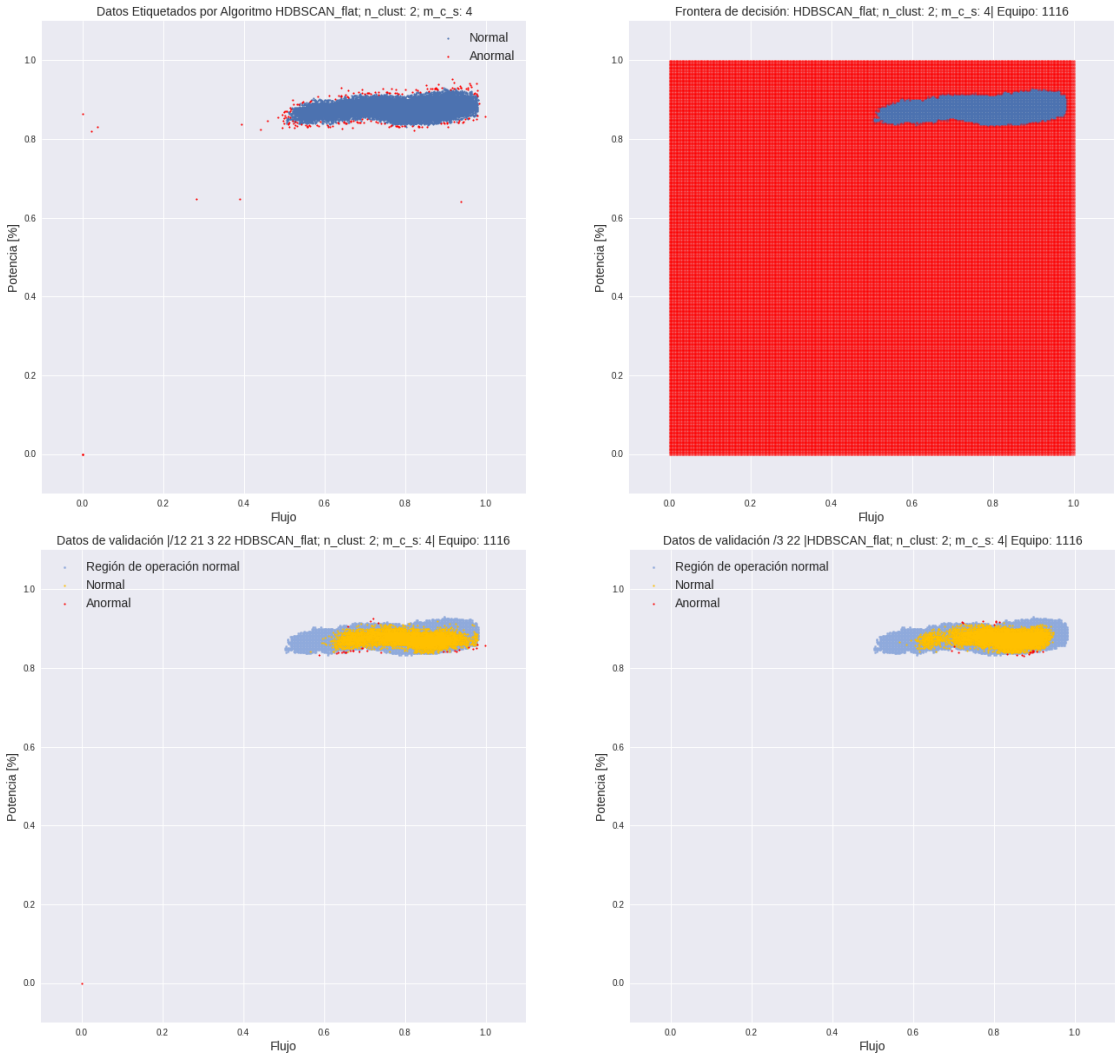

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

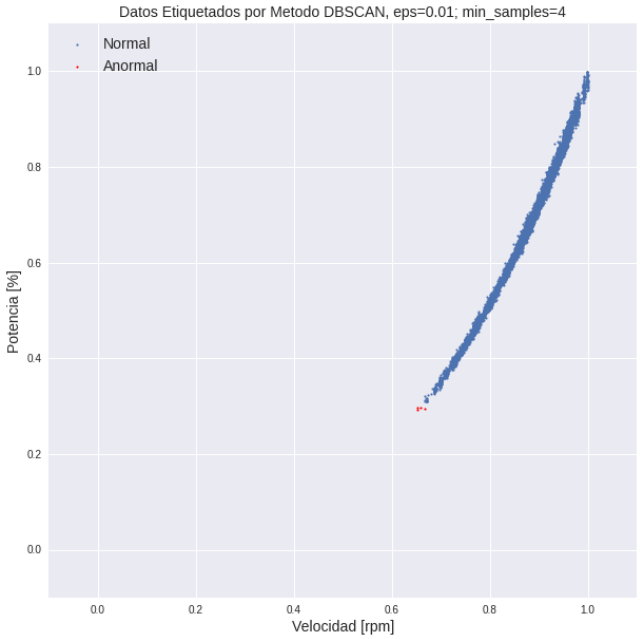

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

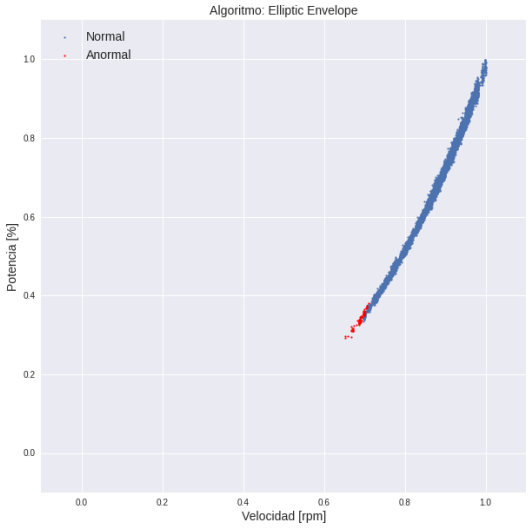

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

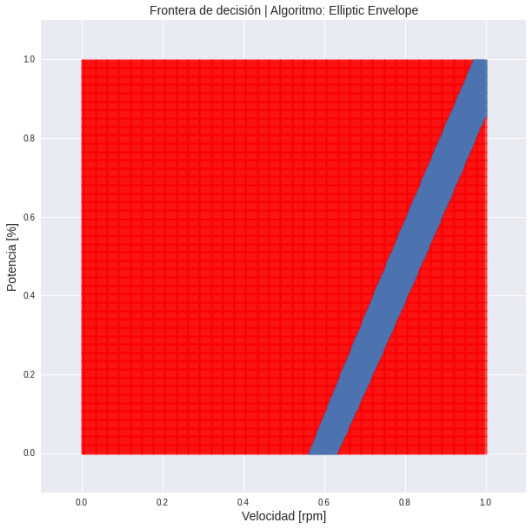
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0085	1.0000	0.0169
Isolation Forest	0.9905	0.0091	1.0000	0.0182
Local Outlier Factor	0.9949	0.0167	1.0000	0.0328
OC-SVM	0.9899	0.0085	1.0000	0.0169

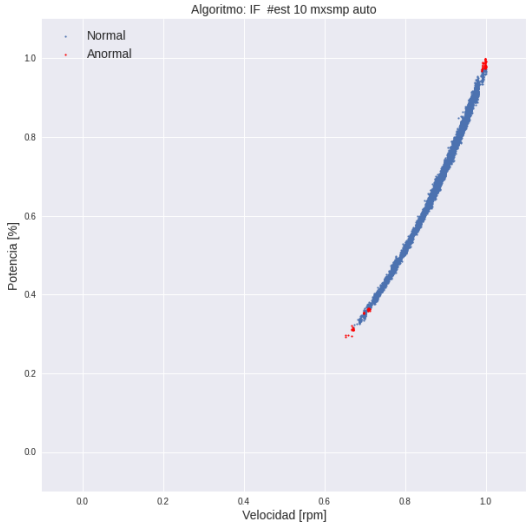


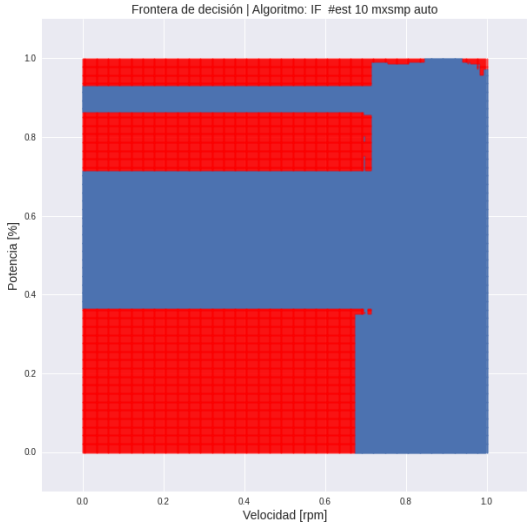
Bomba 1134

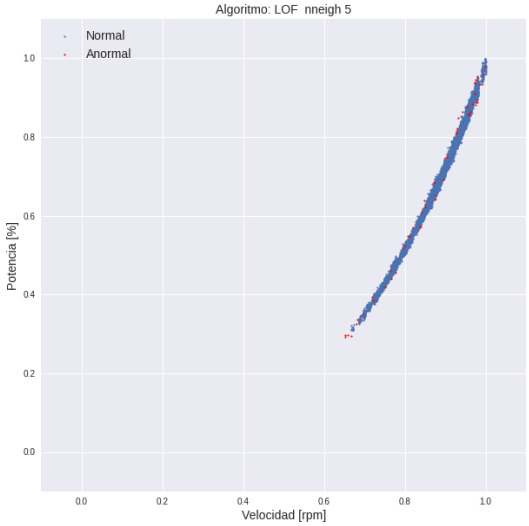

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

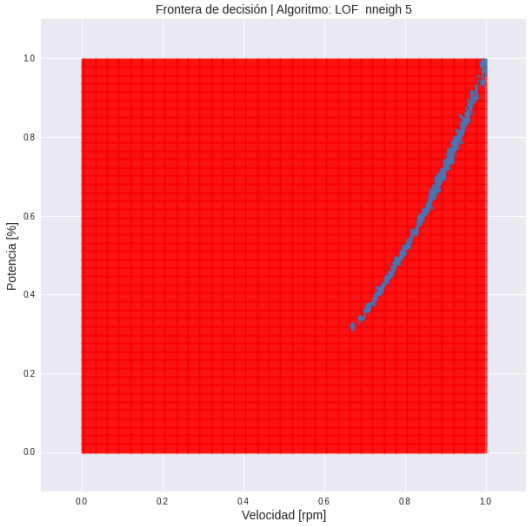

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

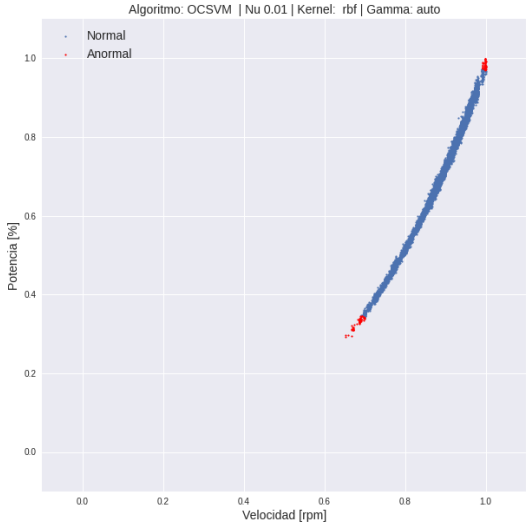

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

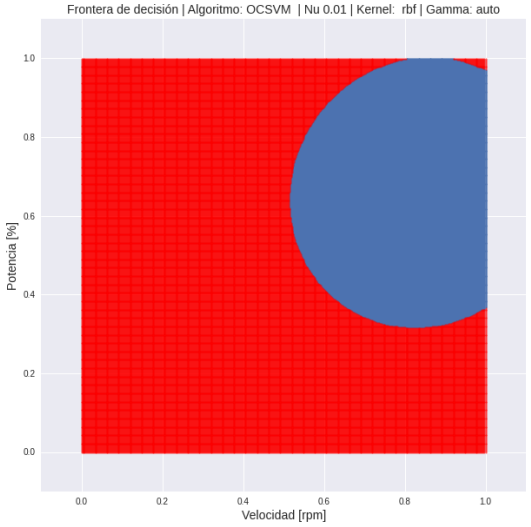

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

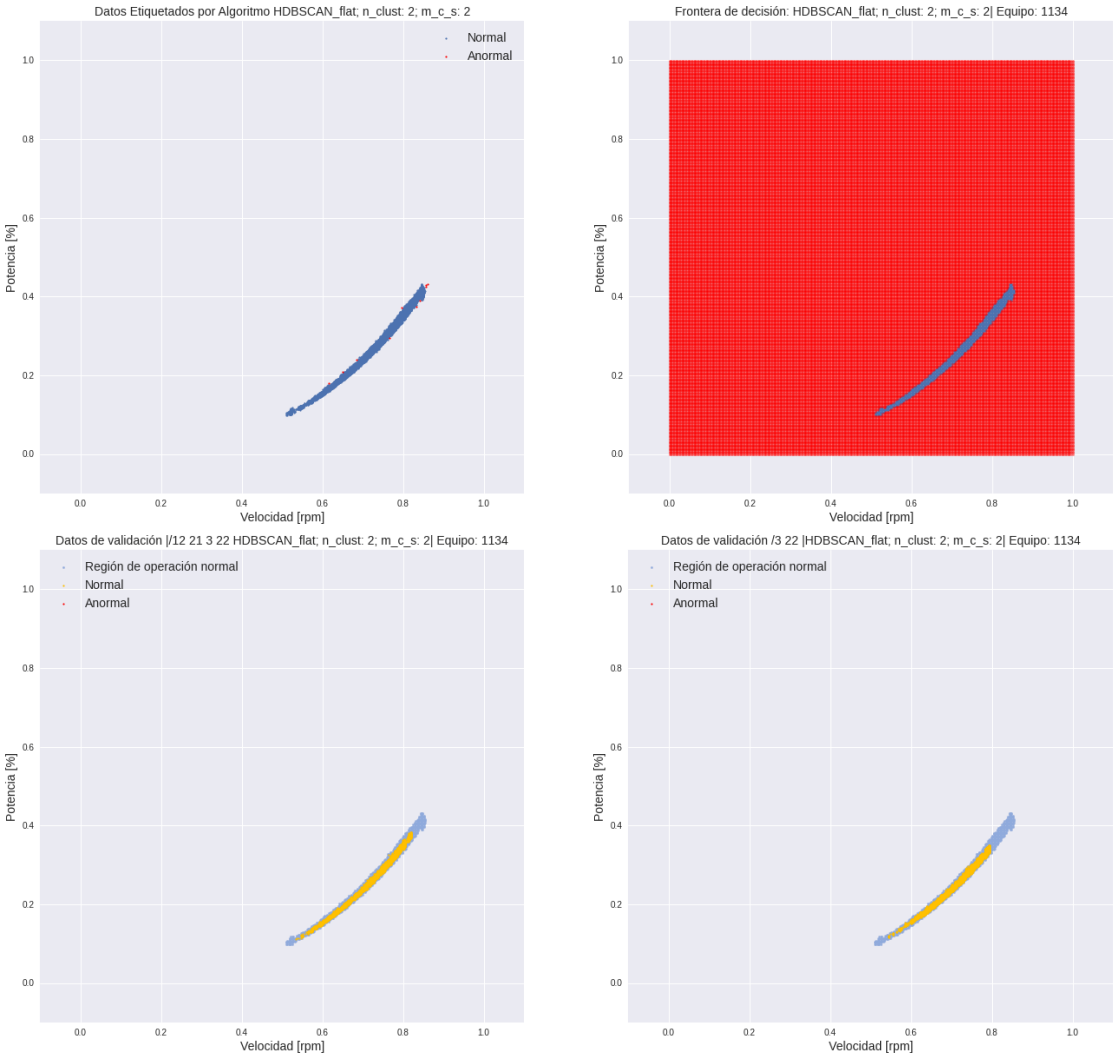

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

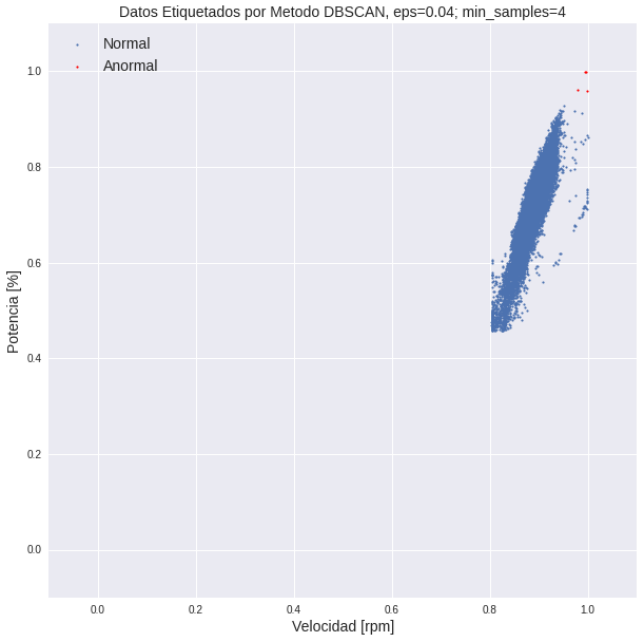

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

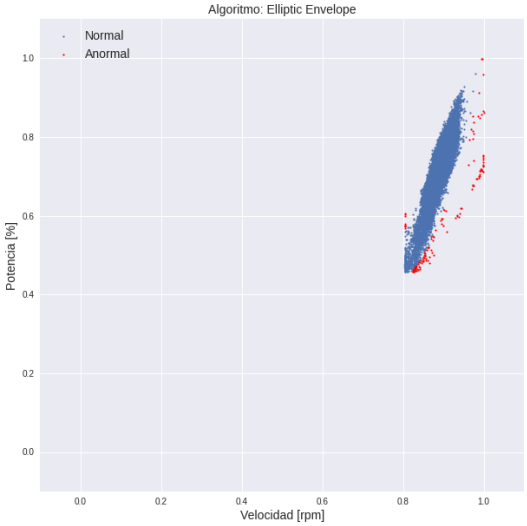

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

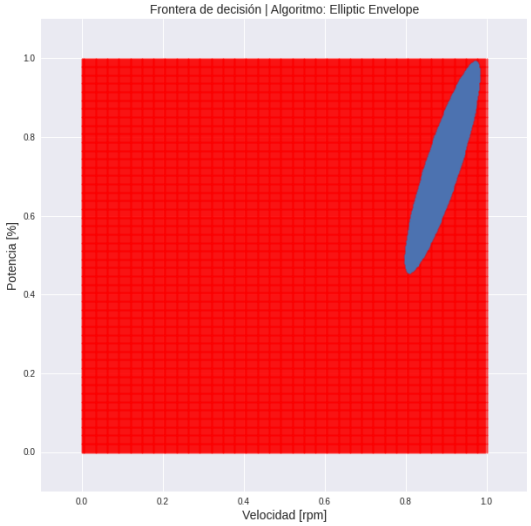
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0333	1.0000	0.0645
Isolation Forest	0.9925	0.0444	1.0000	0.0851
Local Outlier Factor	0.9957	0.0755	1.0000	0.1404
OC-SVM	0.9900	0.0336	1.0000	0.0650

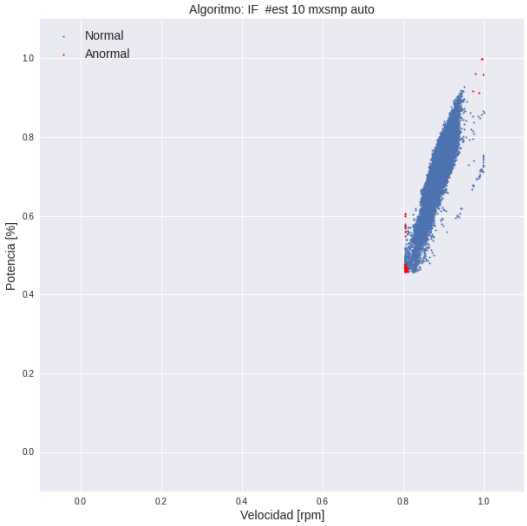


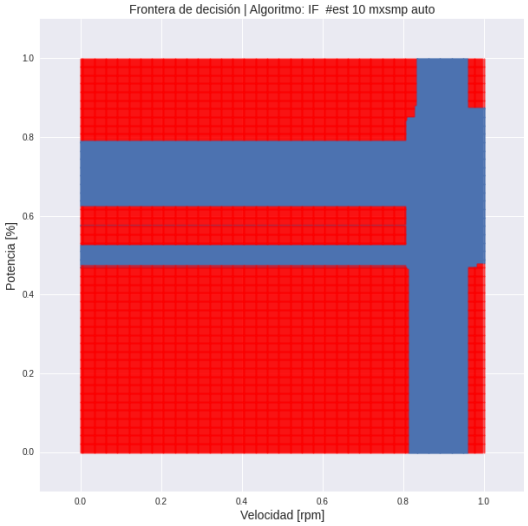
Bomba 1202

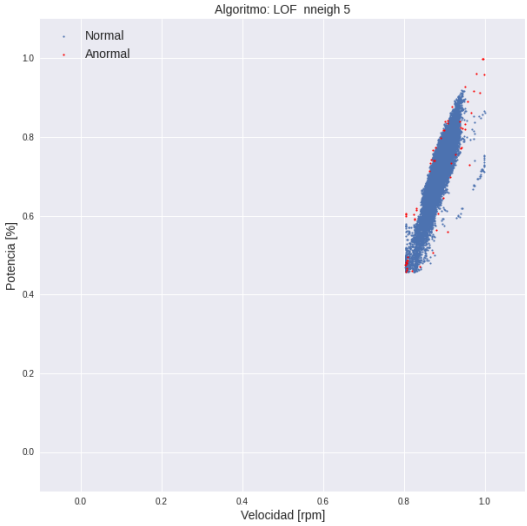

Datos Etiquetados por Metodo DBSCAN, eps=0.04; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]

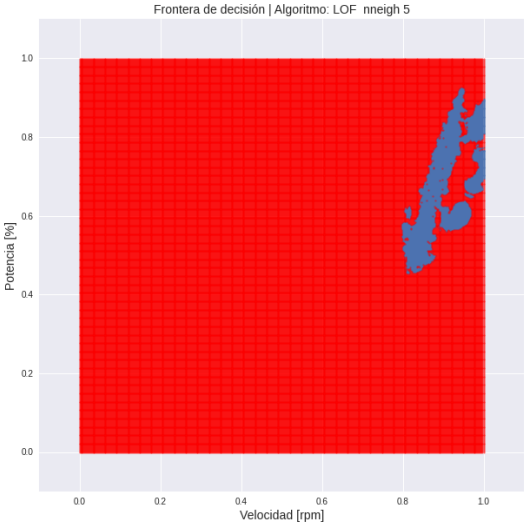

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]

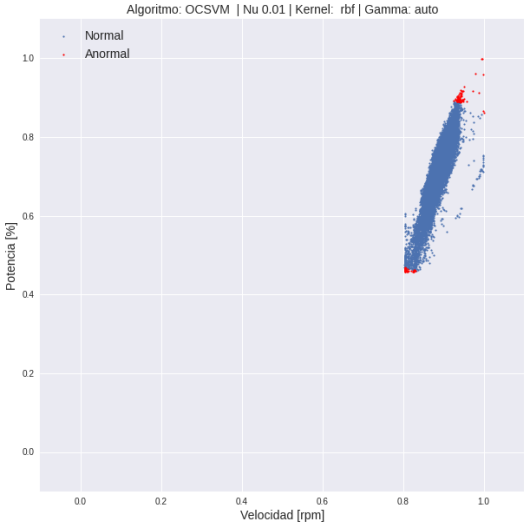

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]

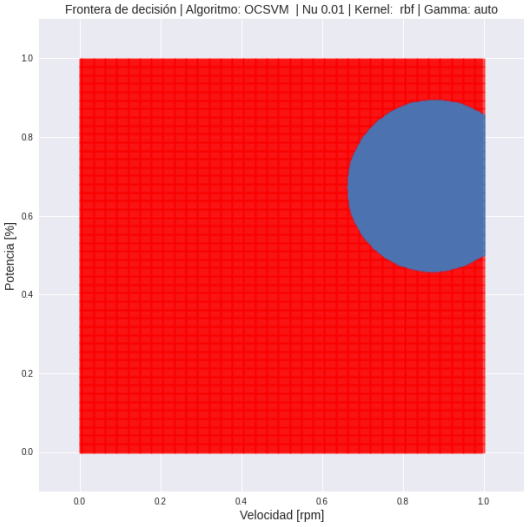

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

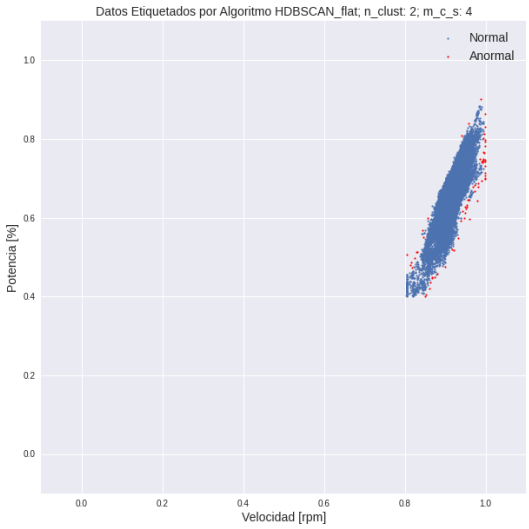

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

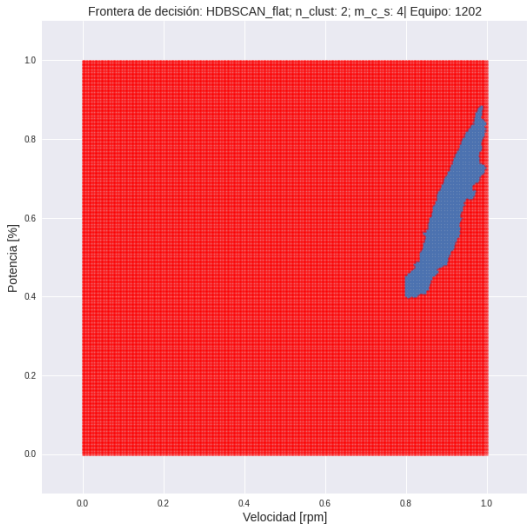

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

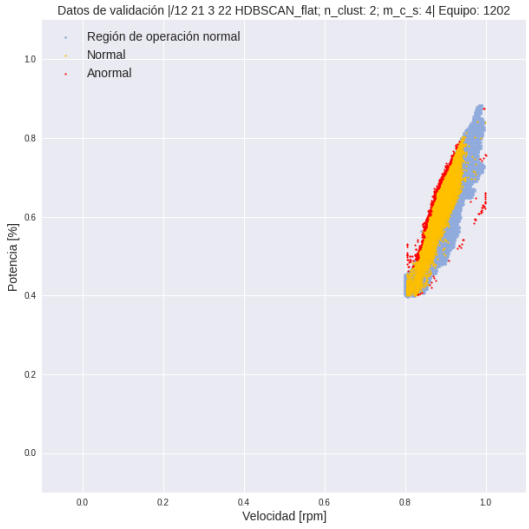

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

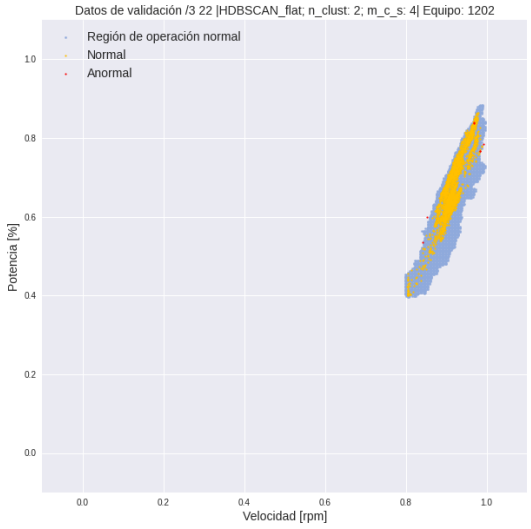

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0234	0.7500	0.4545
Isolation Forest	0.9945	0.0519	1.0000	0.0988
Local Outlier Factor	0.9955	0.0666	1.0000	0.1250
OC-SVM	0.9900	0.0313	1.0000	0.0610

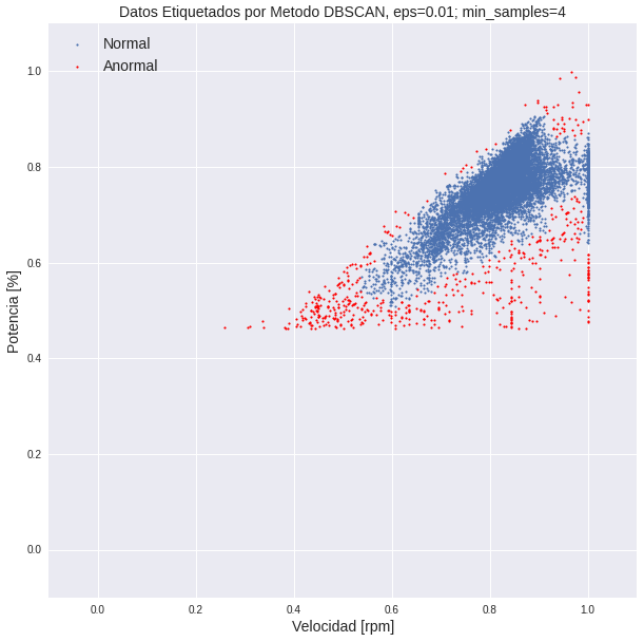


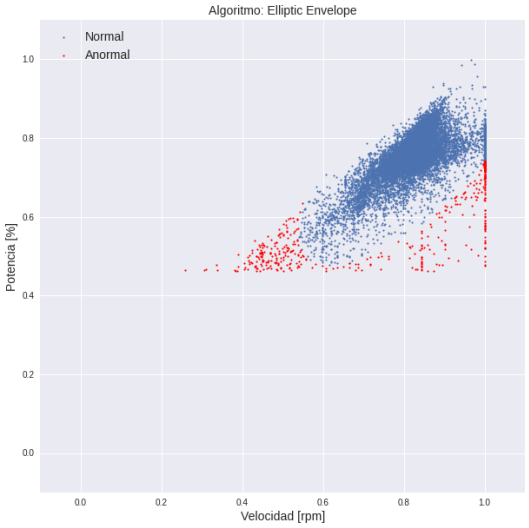


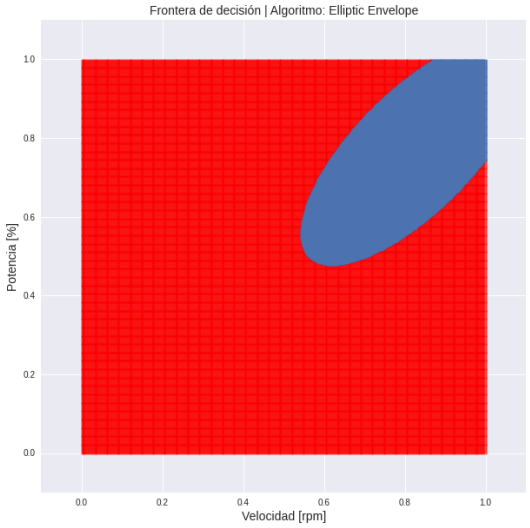


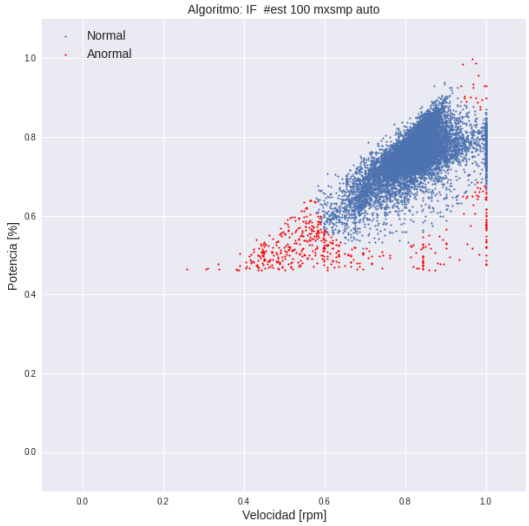


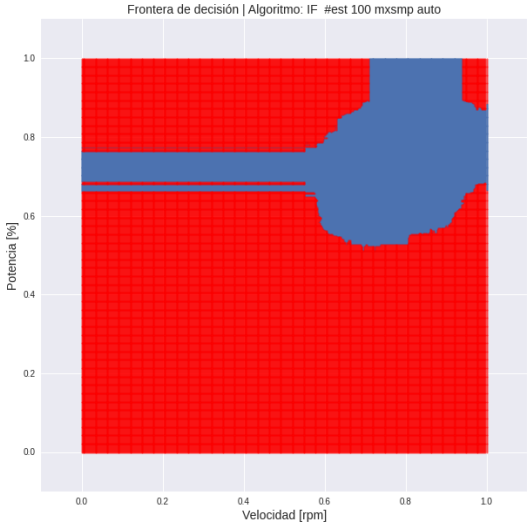
Bomba 1211

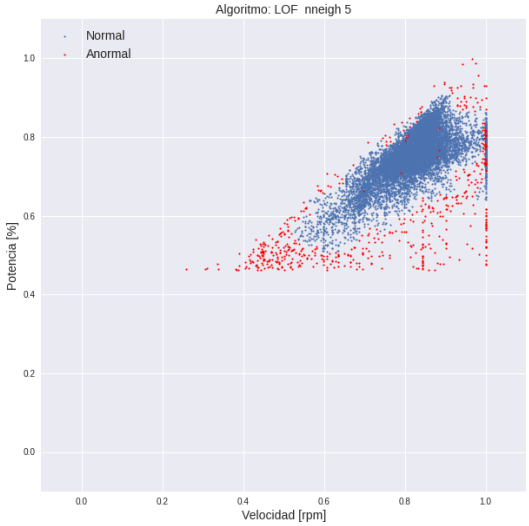

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

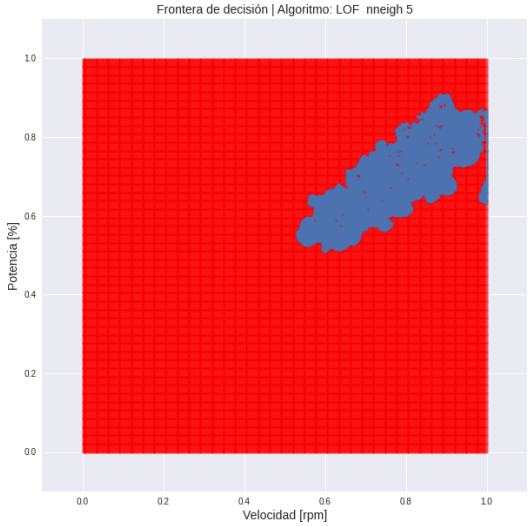

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

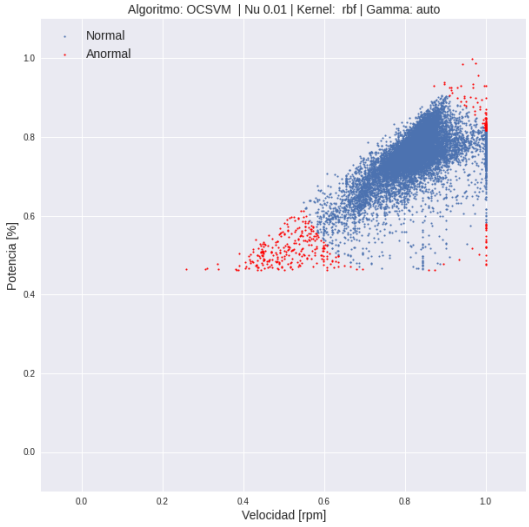

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

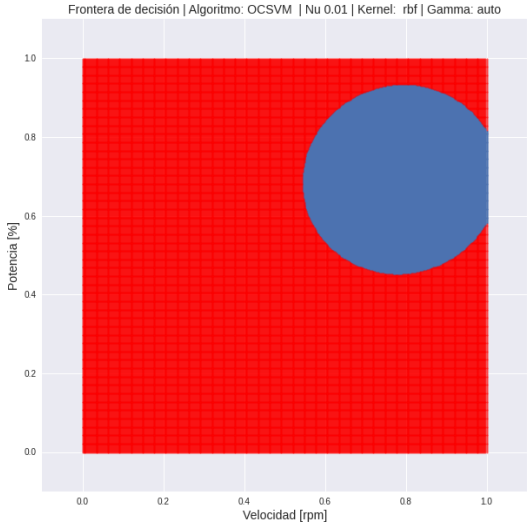

Algoritmo: IF #est 100 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 100 mxsmp auto
Potencia [%]
Velocidad [rpm]

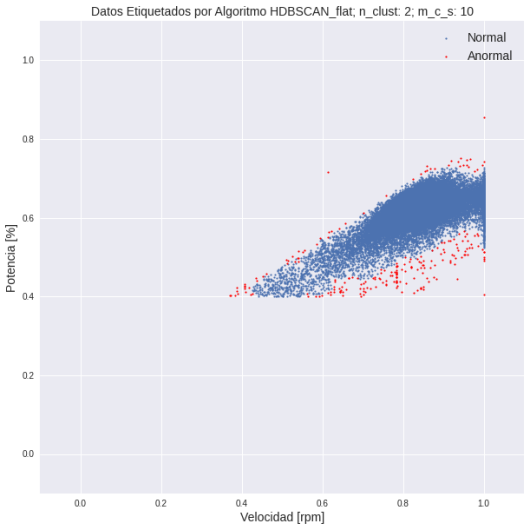

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

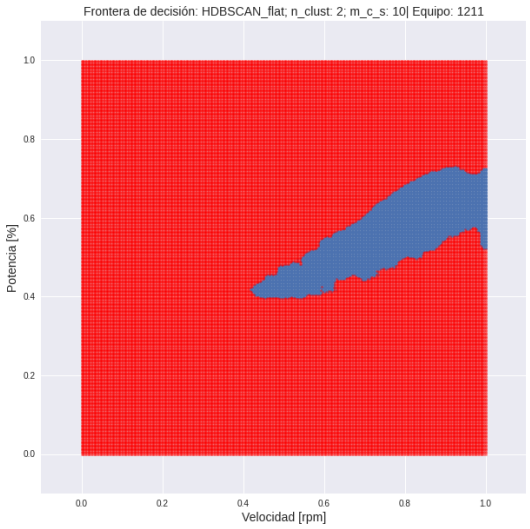

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

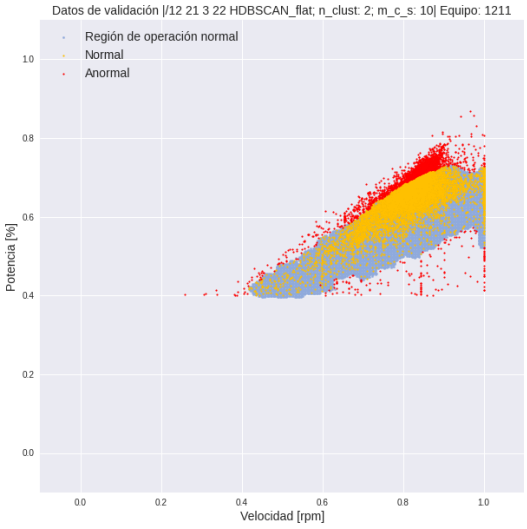

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

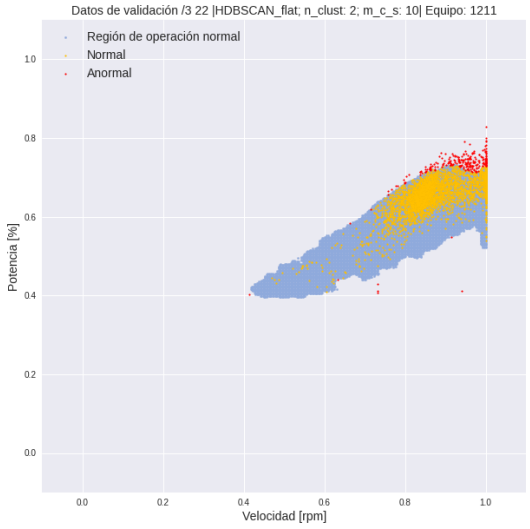

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9739	0.7387	0.6236	0.6763
Isolation Forest	0.9774	0.7608	0.7015	0.7299
Local Outlier Factor	0.9918	0.8806	0.9392	0.9089
OC-SVM	0.9698	0.7066	0.5266	0.6035

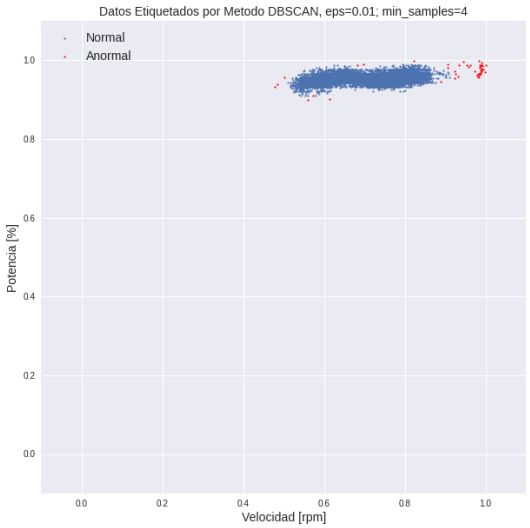


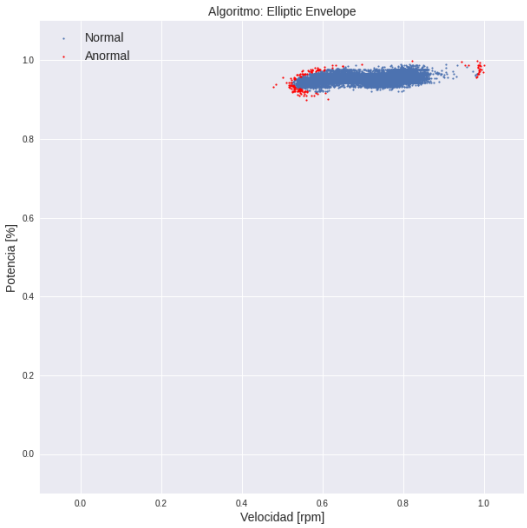


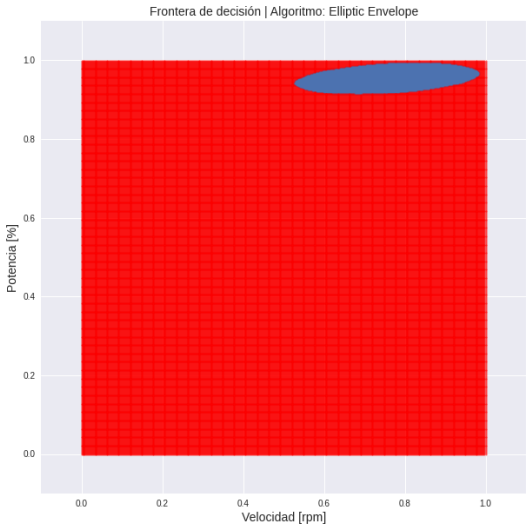


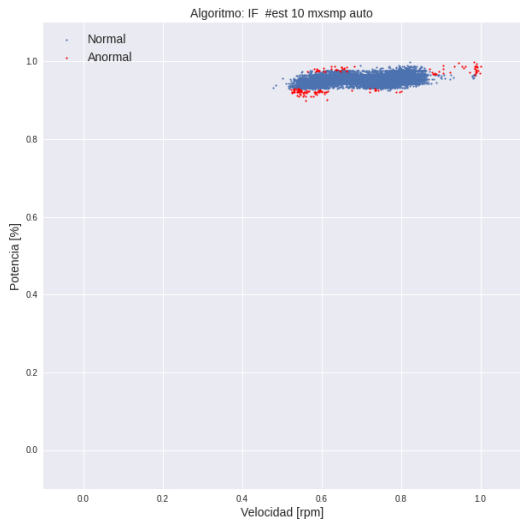


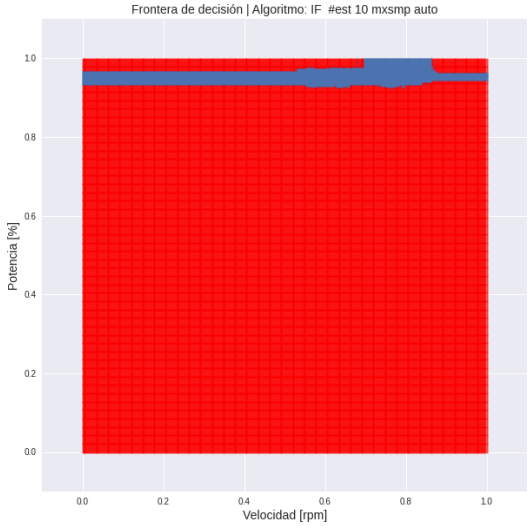
Bomba 2116

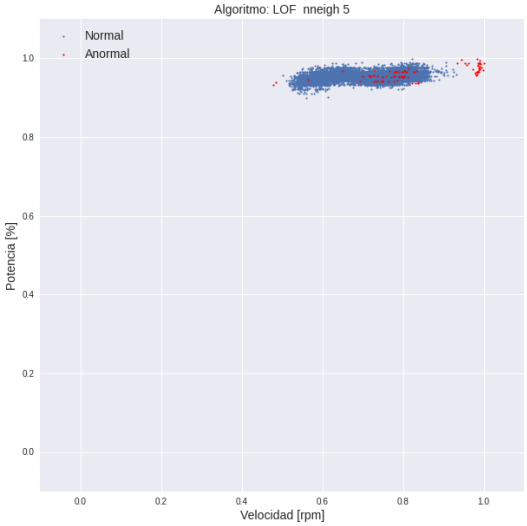

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]

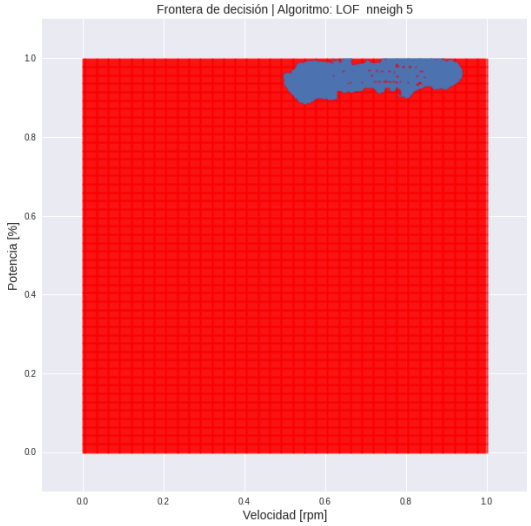

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]

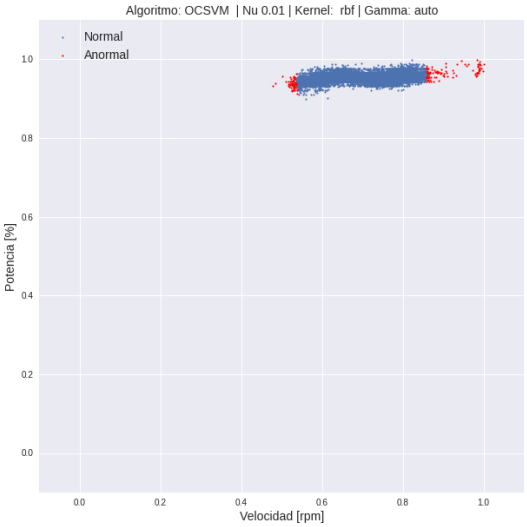

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]

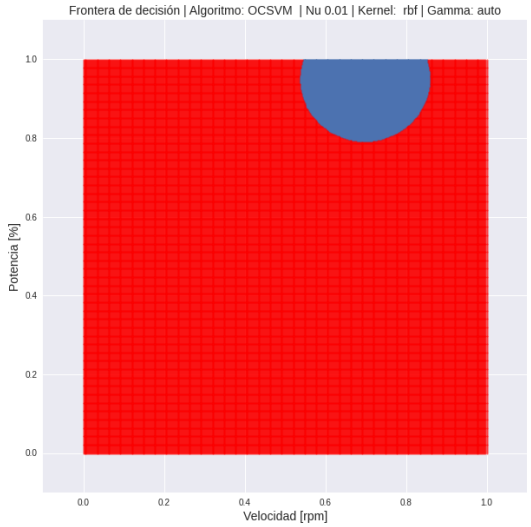

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

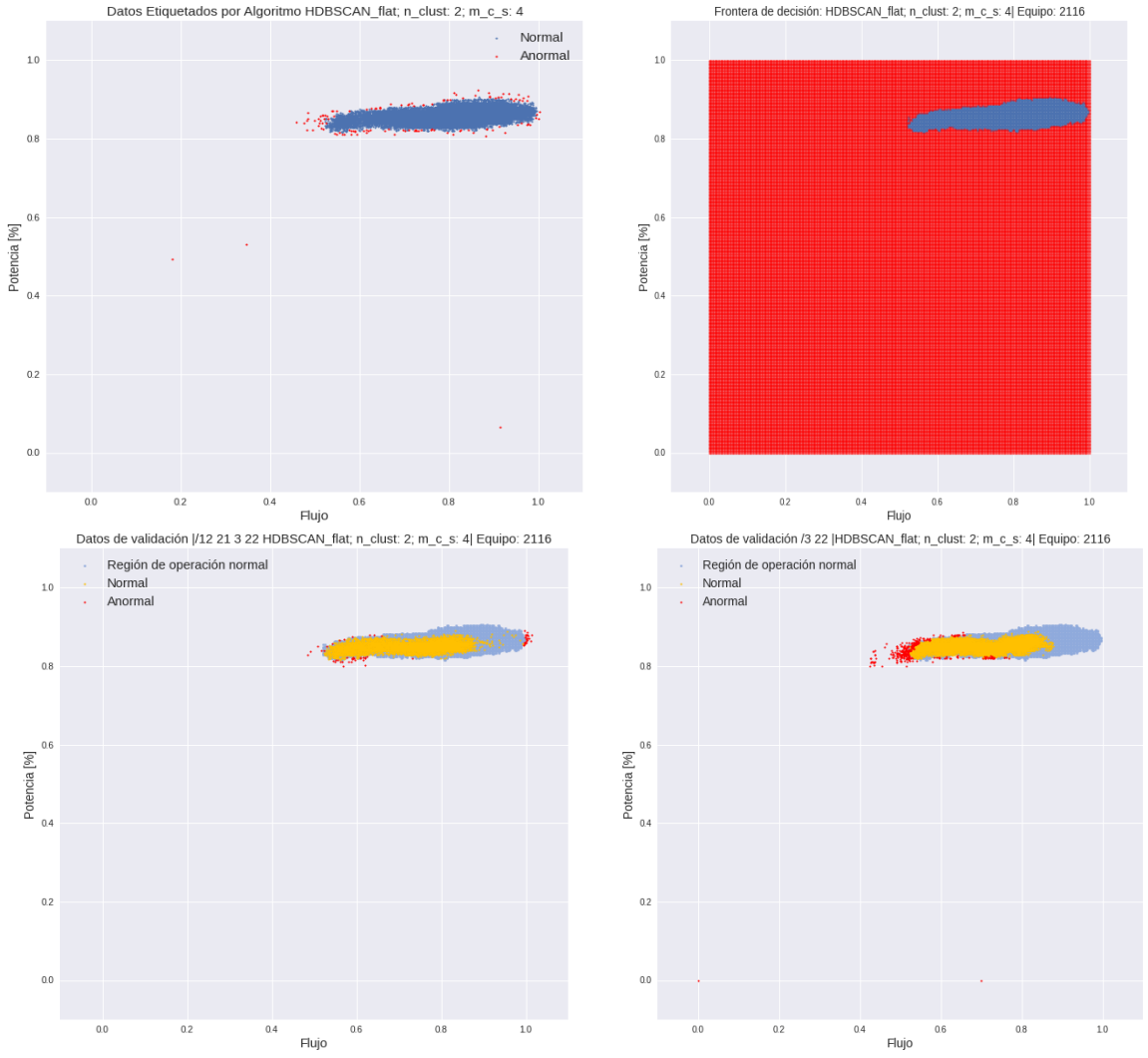

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

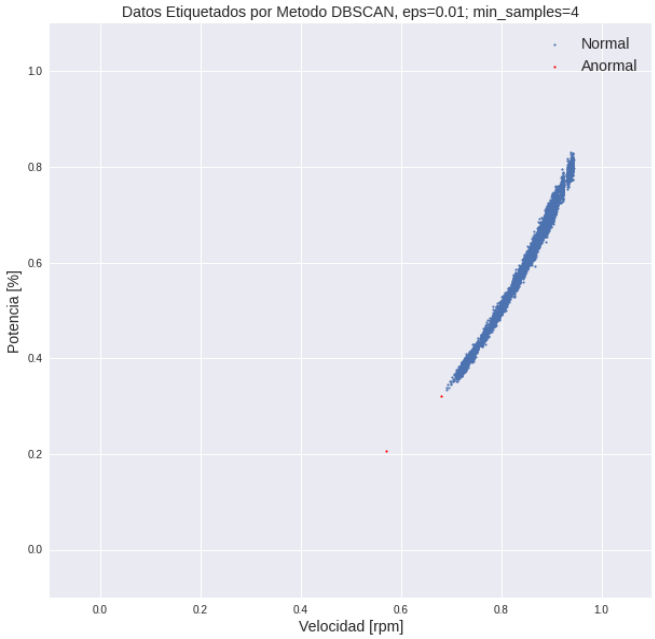

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

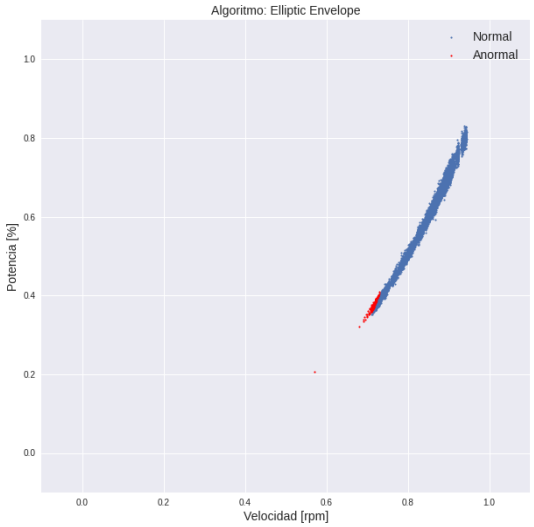

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

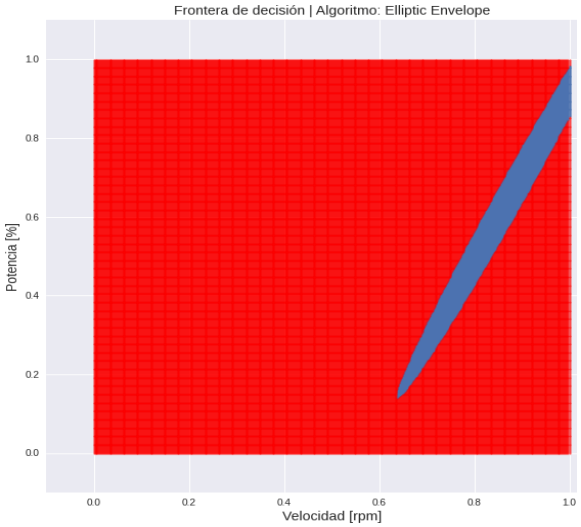
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9887	0.2157	0.6735	0.3267
Isolation Forest	0.9885	0.2053	0.6327	0.3100
Local Outlier Factor	0.9944	0.3977	0.7143	0.5109
OC-SVM	0.9896	0.2671	0.8775	0.4095

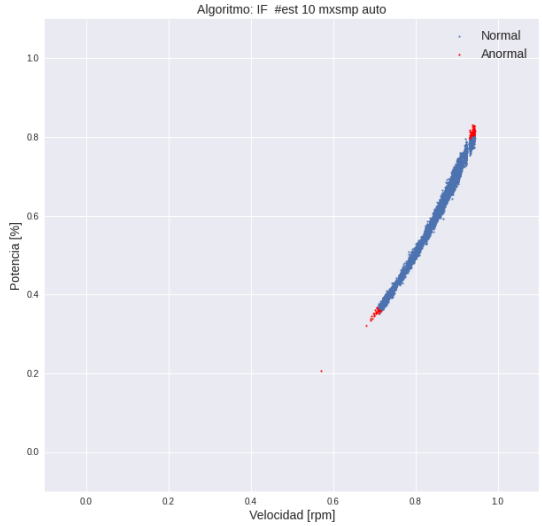


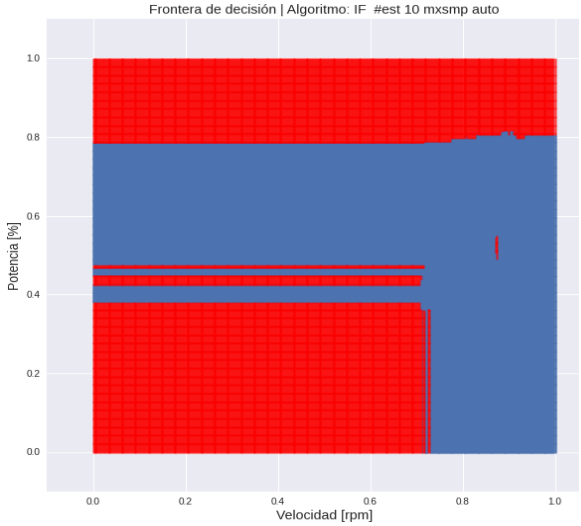
Bomba 2134

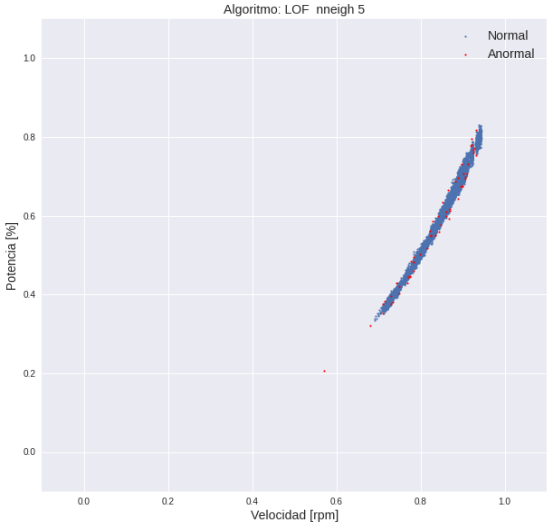

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

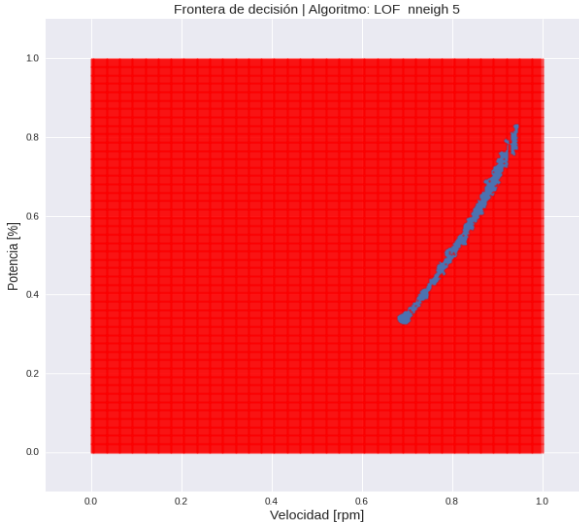

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

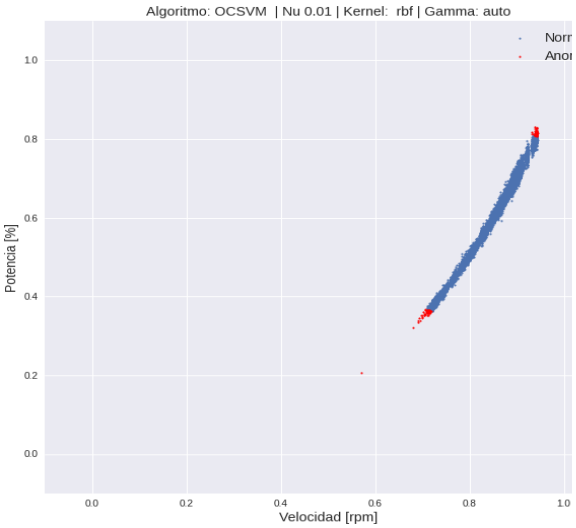

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

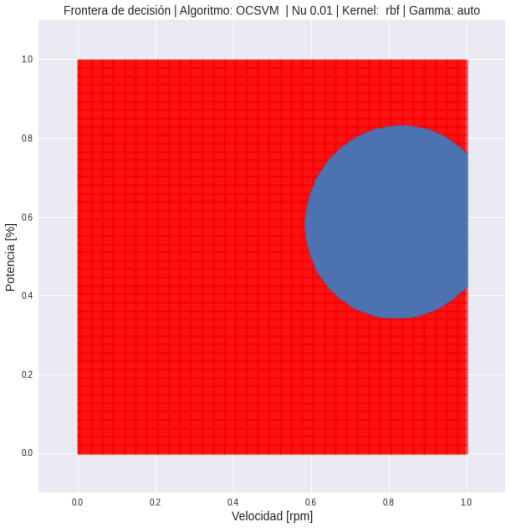

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

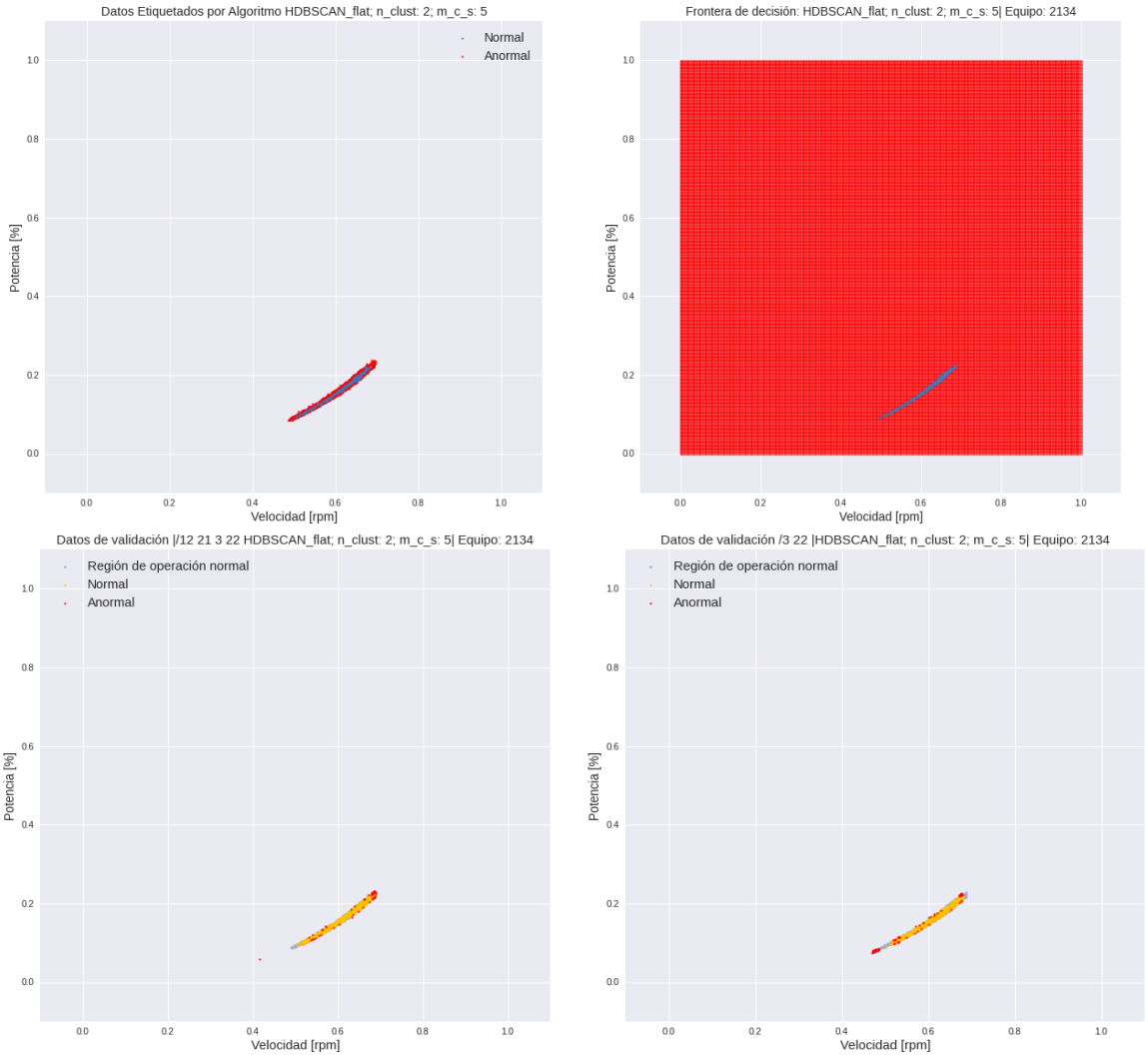

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

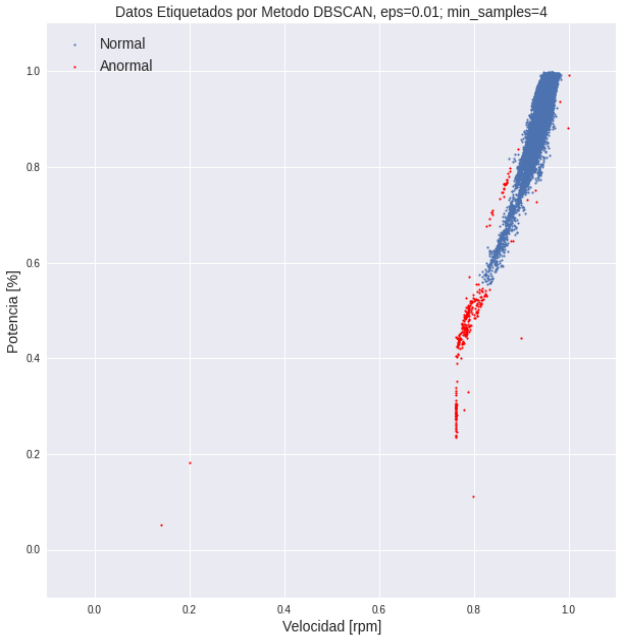

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

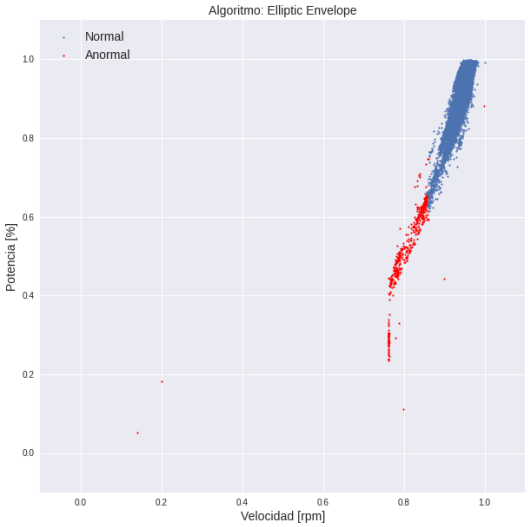

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

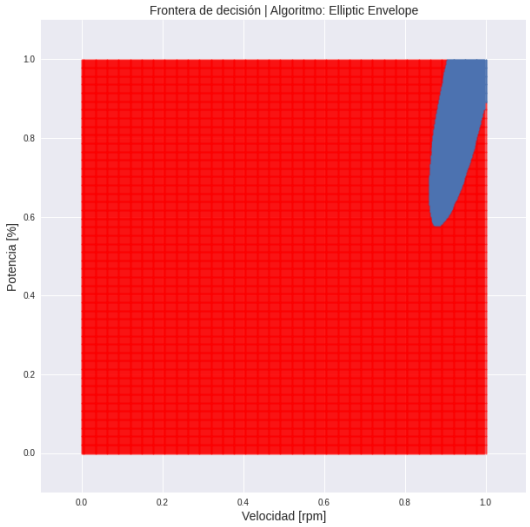
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0164	1.0000	0.0323
Isolation Forest	0.9906	0.0175	1.0000	0.0345
Local Outlier Factor	0.9955	0.0357	1.0000	0.0689
OC-SVM	0.9899	0.0164	1.0000	0.0323

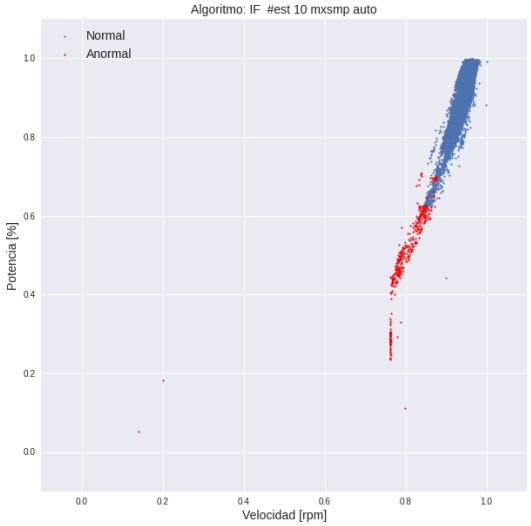


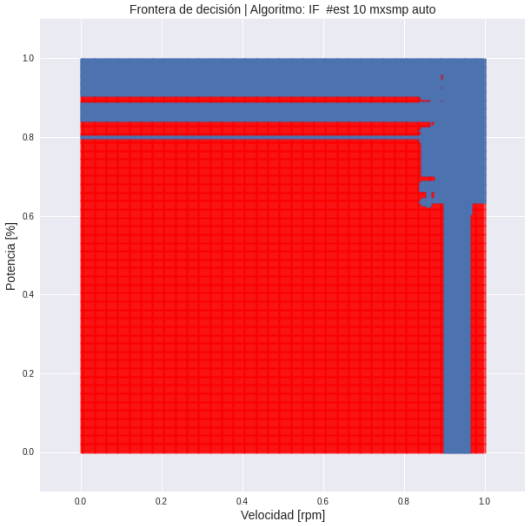
Bomba 2202

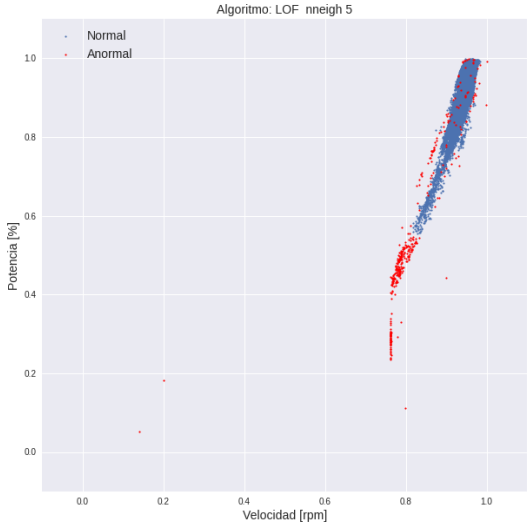

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

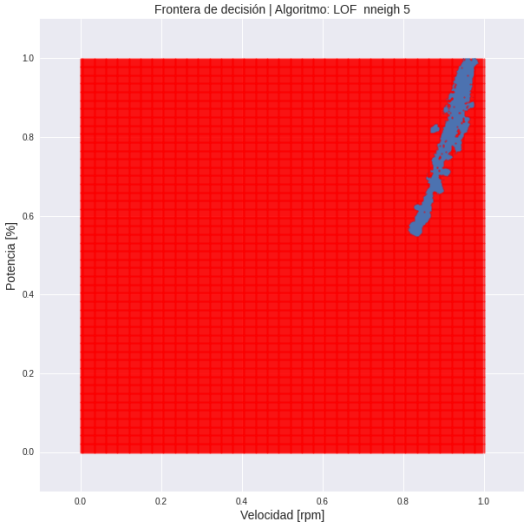

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

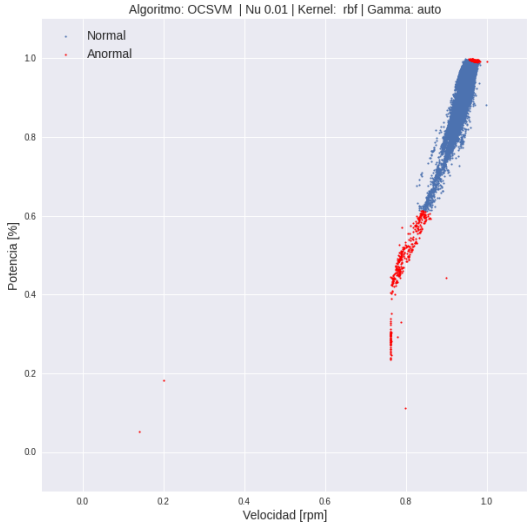

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

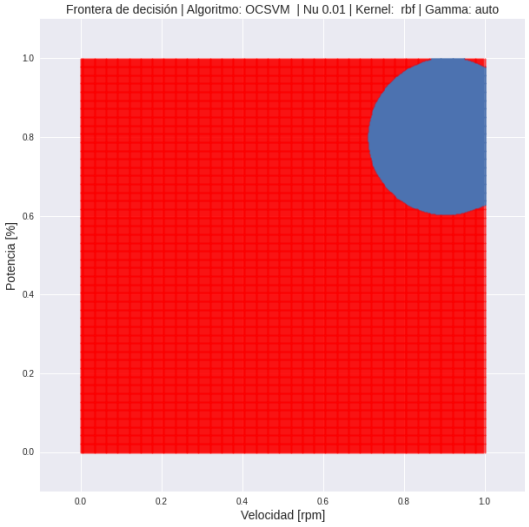

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

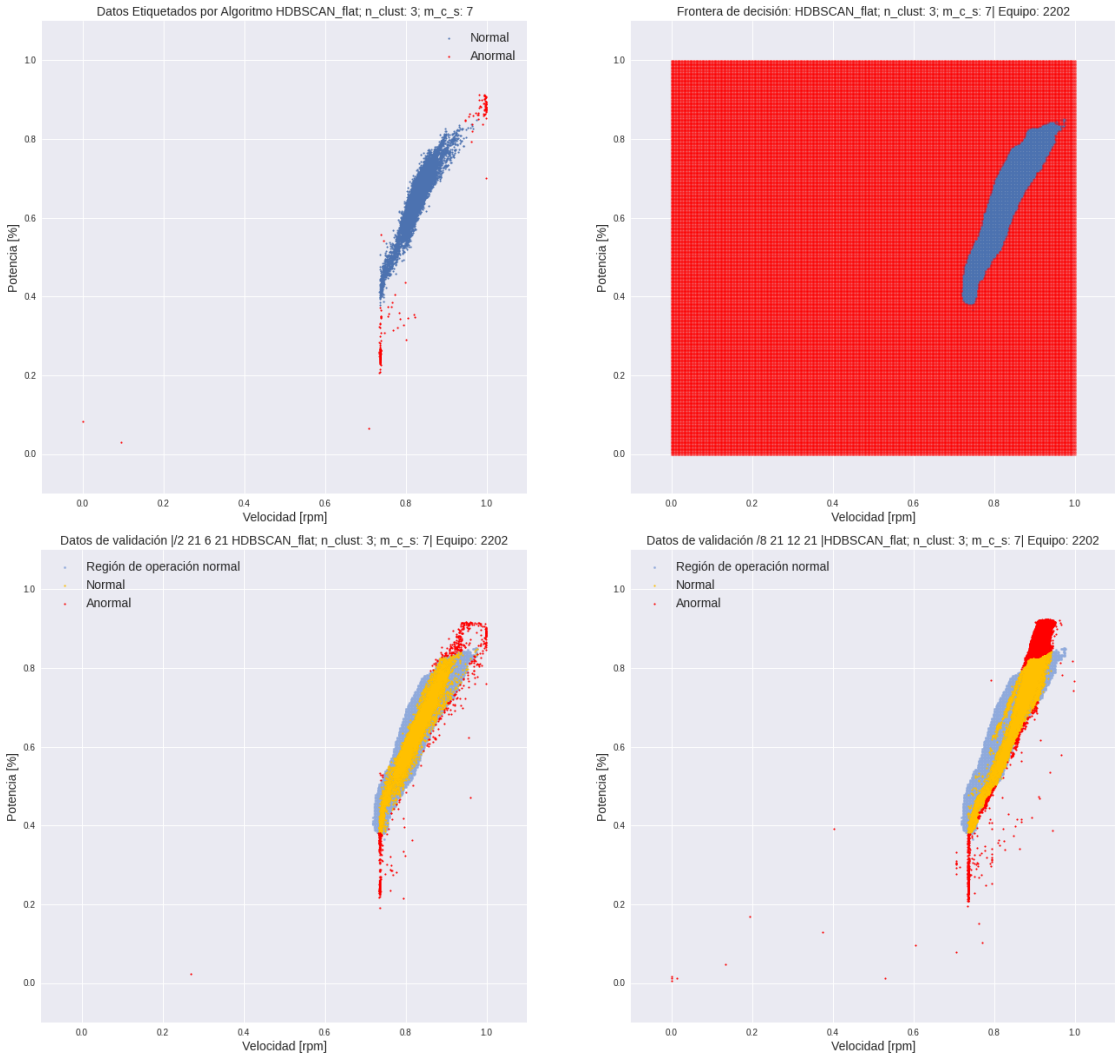

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

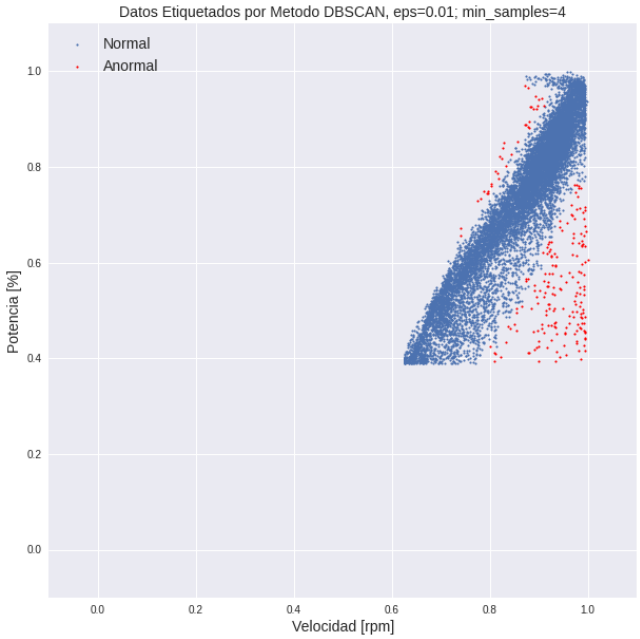

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

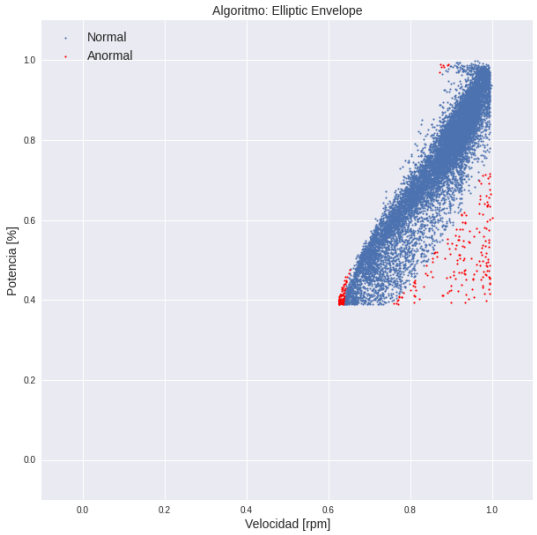

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

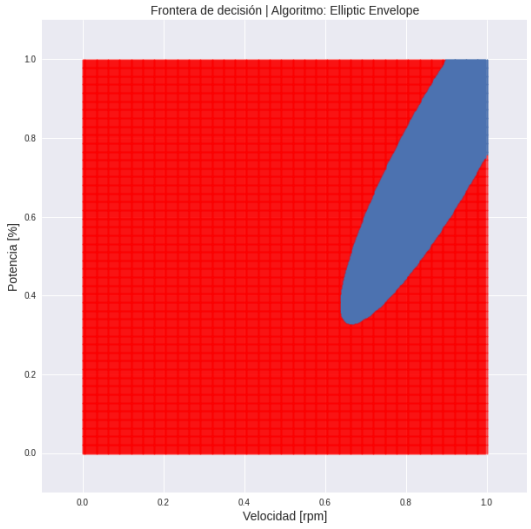
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9888	0.5846	0.9161	0.7137
Isolation Forest	0.9905	0.6321	0.8993	0.7424
Local Outlier Factor	0.9959	0.7905	1.0000	0.8829
OC-SVM	0.9884	0.5767	0.8826	0.6976

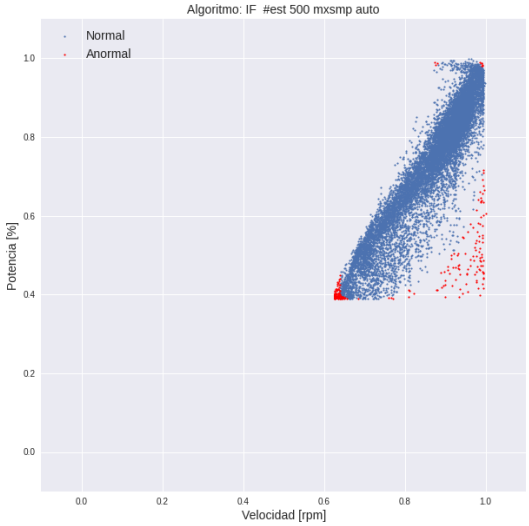


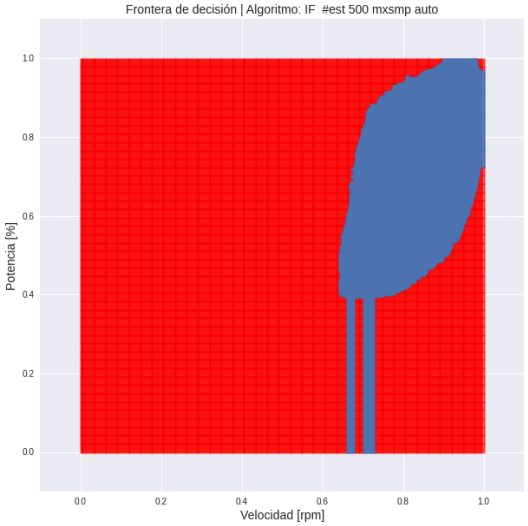
Bomba 2211

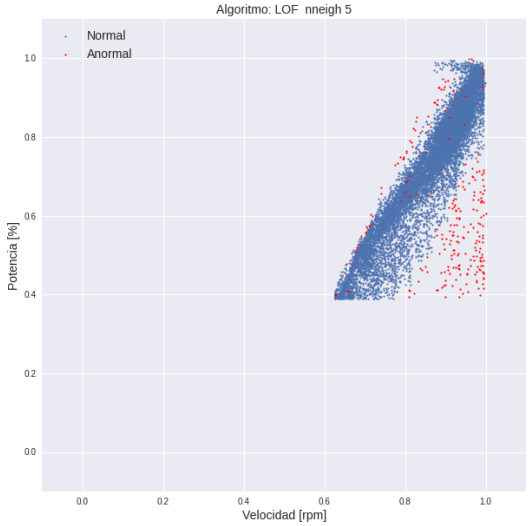

Datos Etiquetados por Metodo DBSCAN, eps=0.01; min_samples=4
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

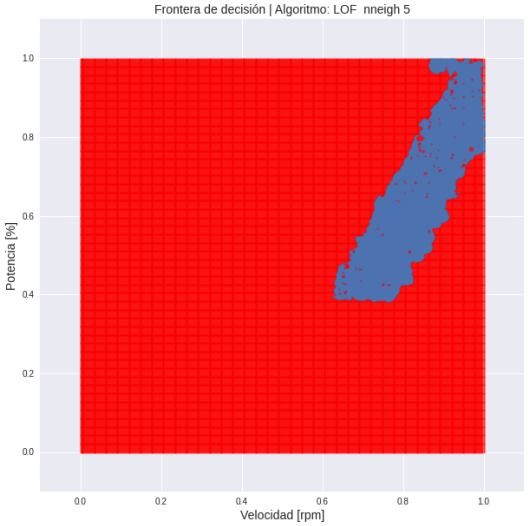

Algoritmo: Elliptic Envelope
Normal
Anormal
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

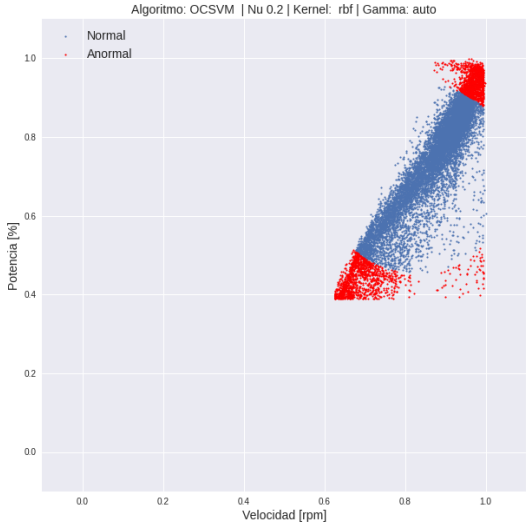

Frontera de decisión | Algoritmo: Elliptic Envelope
Potencia [%]
Velocidad [rpm]
0.0
0.2
0.4
0.6
0.8
1.0

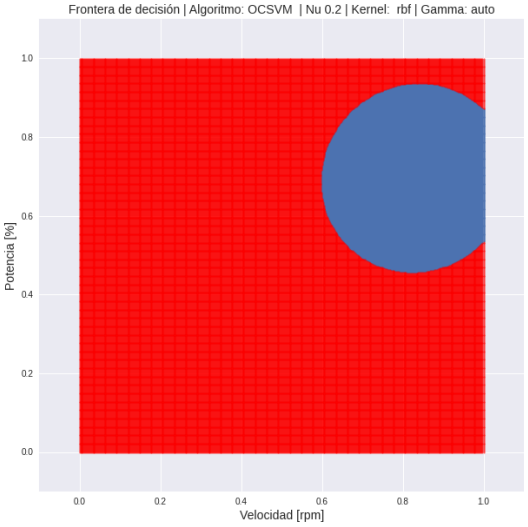

Algoritmo: IF #est 500 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 500 mxsmp auto
Potencia [%]
Velocidad [rpm]

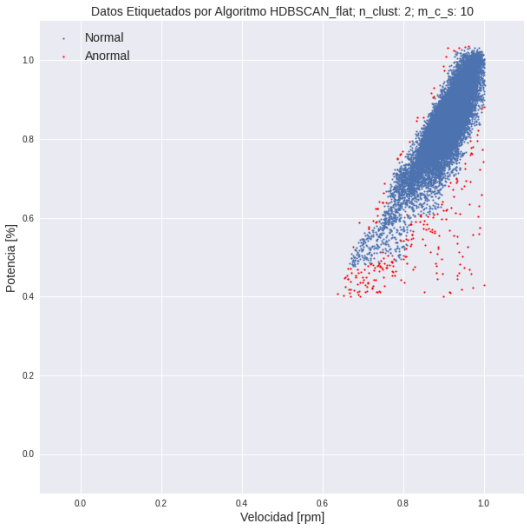

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

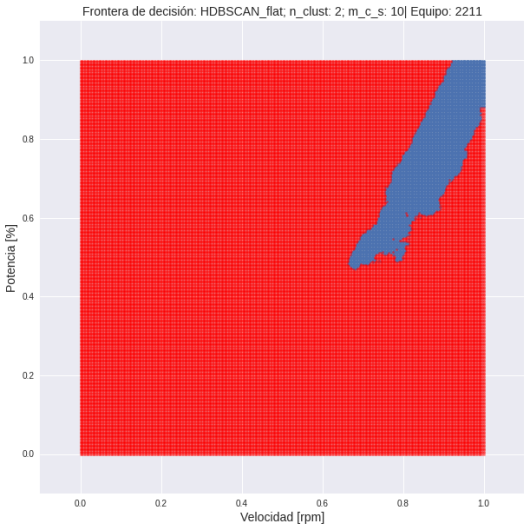

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

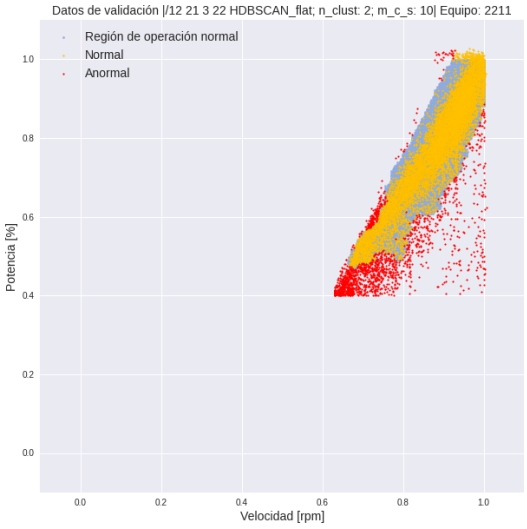

Algoritmo: OCSVM | Nu 0.2 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

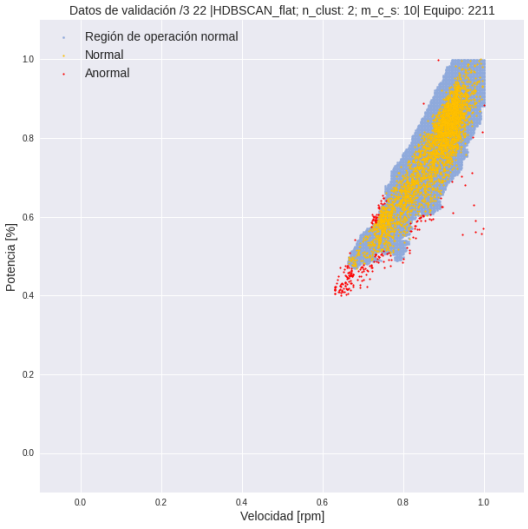

Frontera de decisión | Algoritmo: OCSVM | Nu 0.2 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9853	0.5588	0.7204	0.6294
Isolation Forest	0.9815	0.4667	0.4976	0.4817
Local Outlier Factor	0.9945	0.7951	0.9194	0.8527
OC-SVM	0.7922	0.0288	0.3365	0.0530

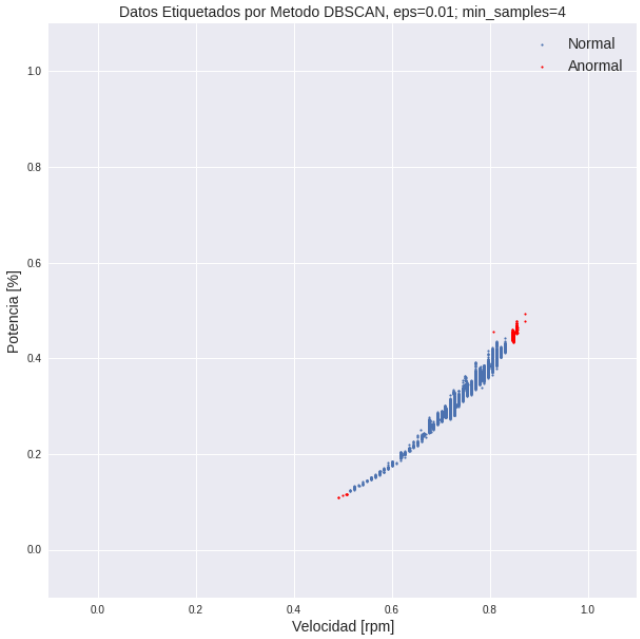


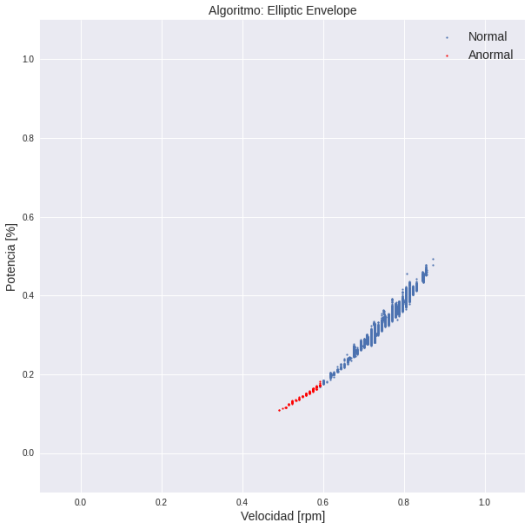


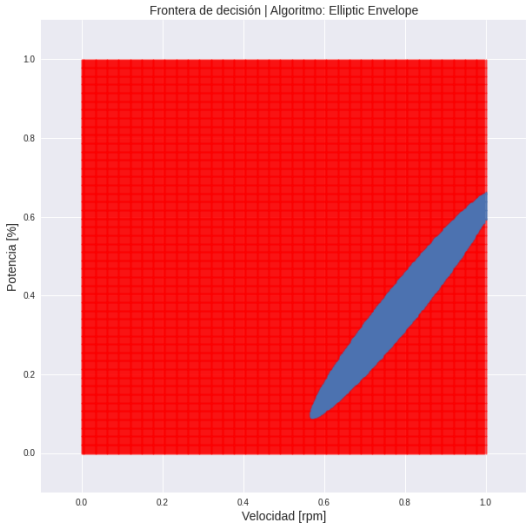


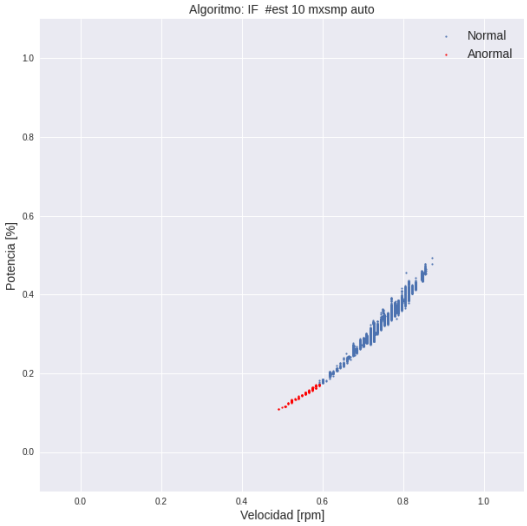


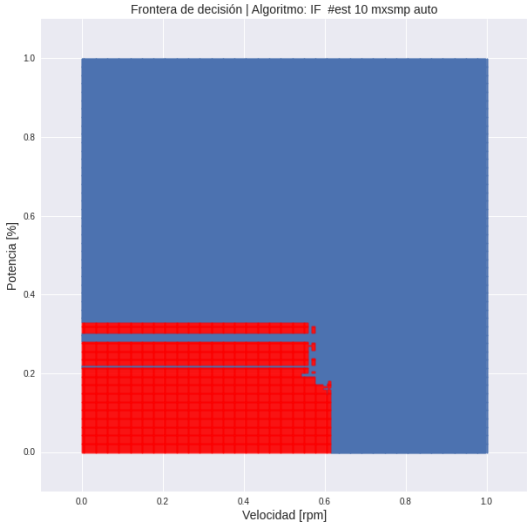
Extractor 1464

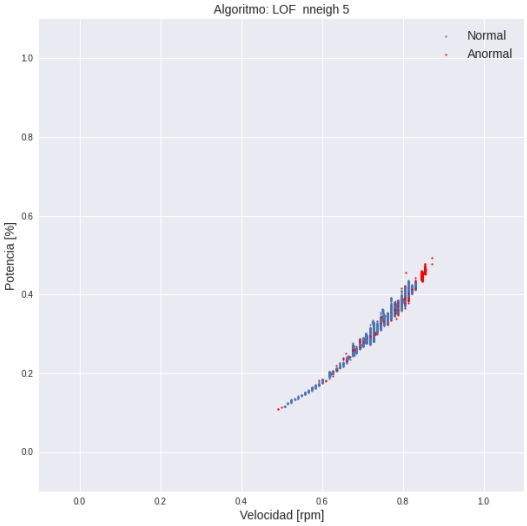


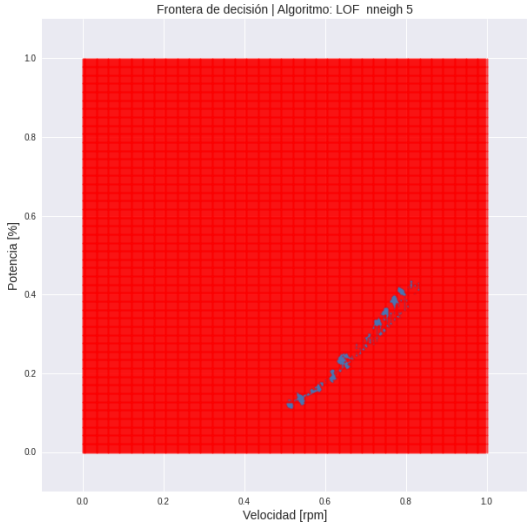


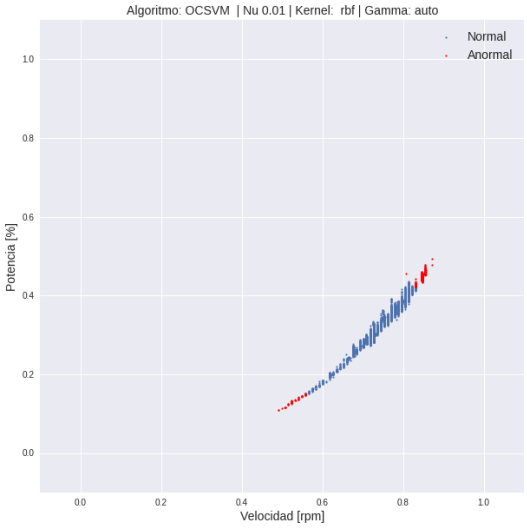


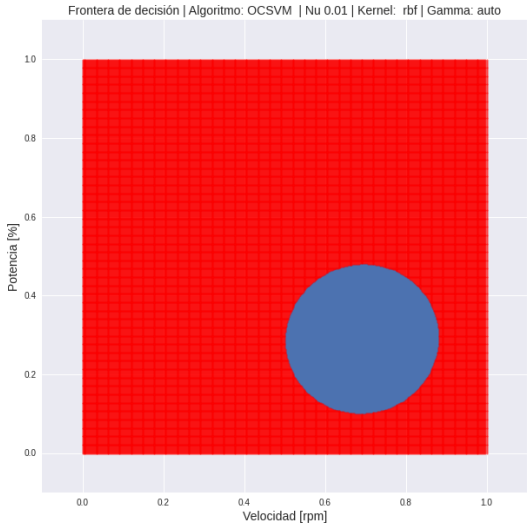

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

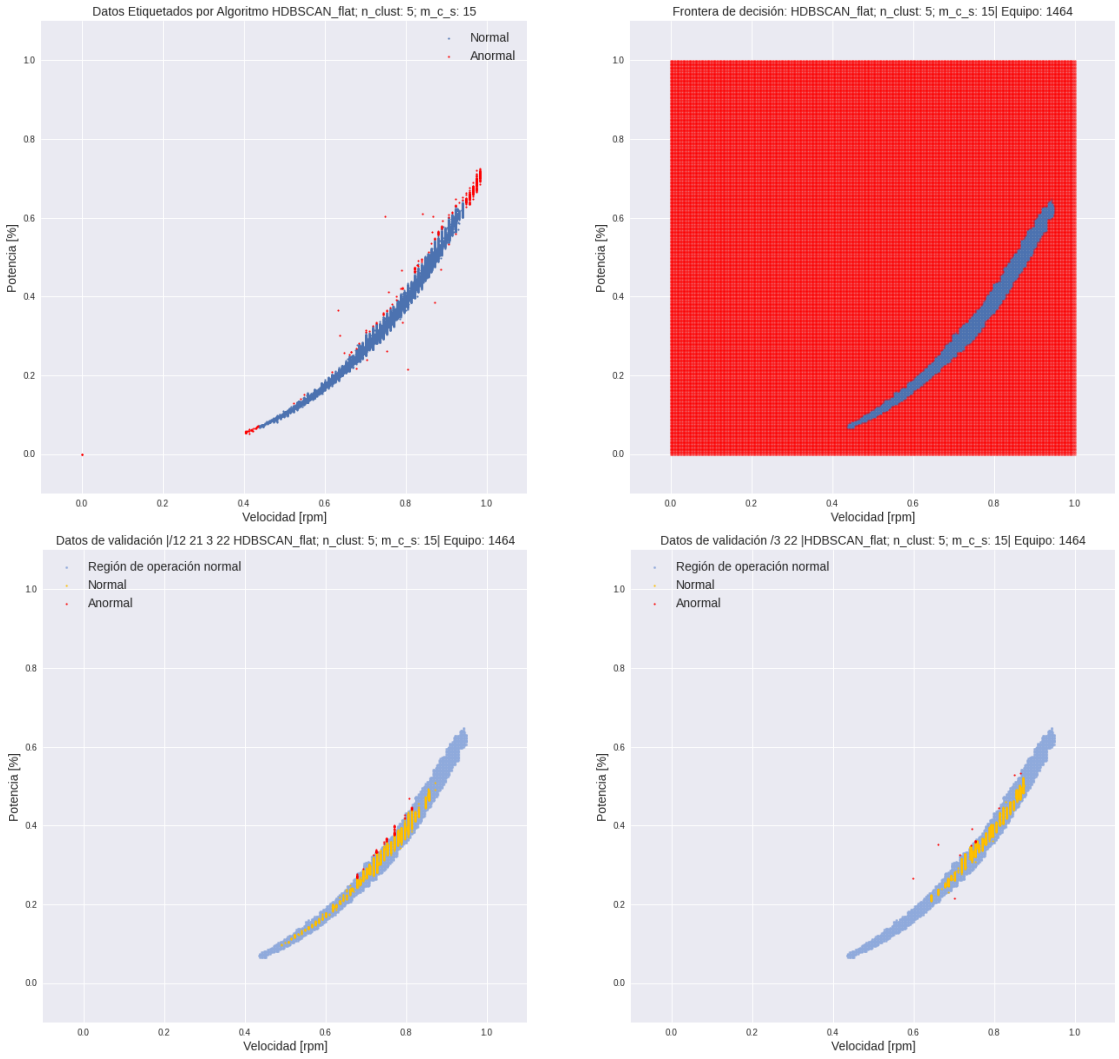

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

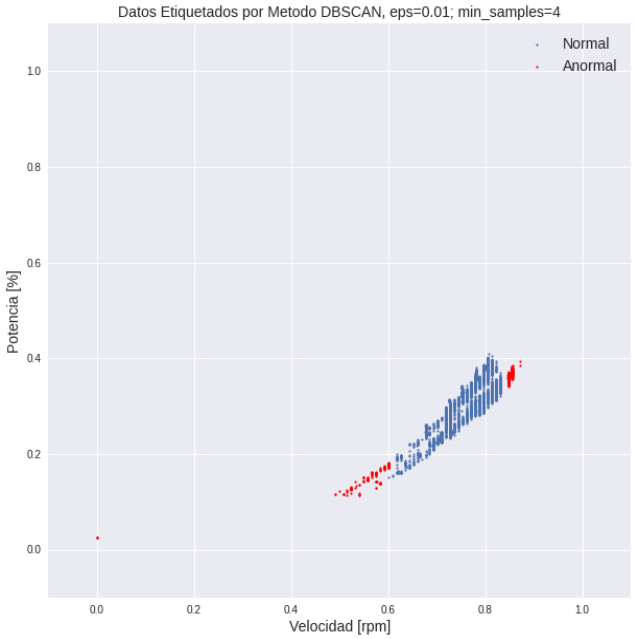

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

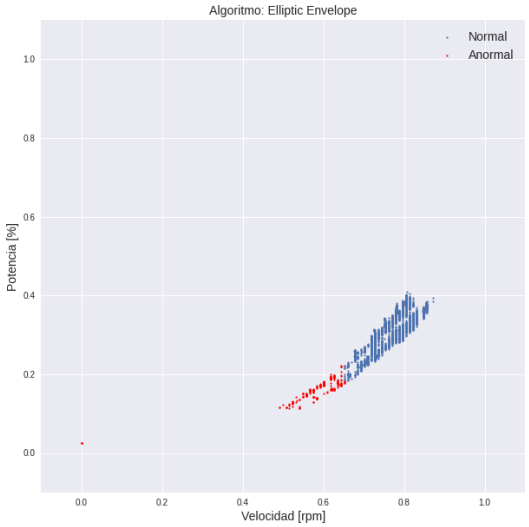

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

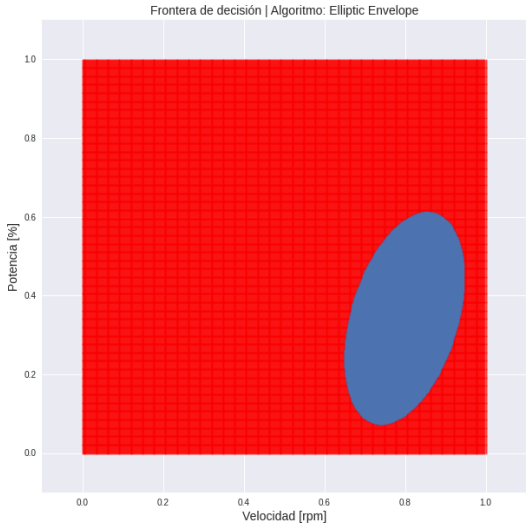
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9382	0.0598	0.0115	0.0193
Isolation Forest	0.9384	0.0614	0.0115	0.0194
Local Outlier Factor	0.9943	0.9070	0.9934	0.9483
OC-SVM	0.9905	0.8482	1.0000	0.9179

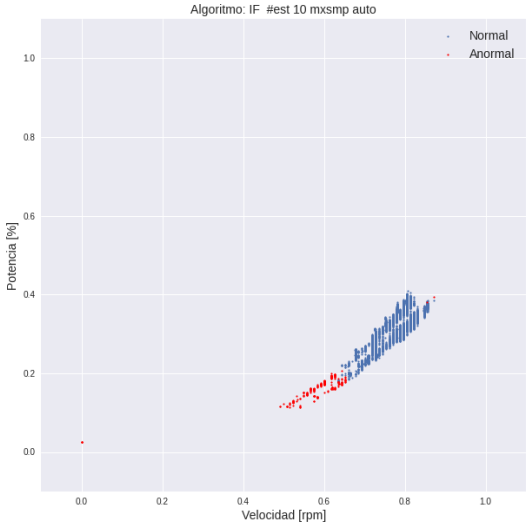


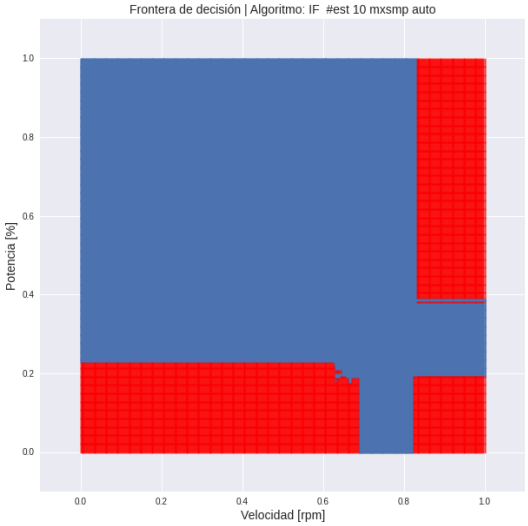
Extractor 1465

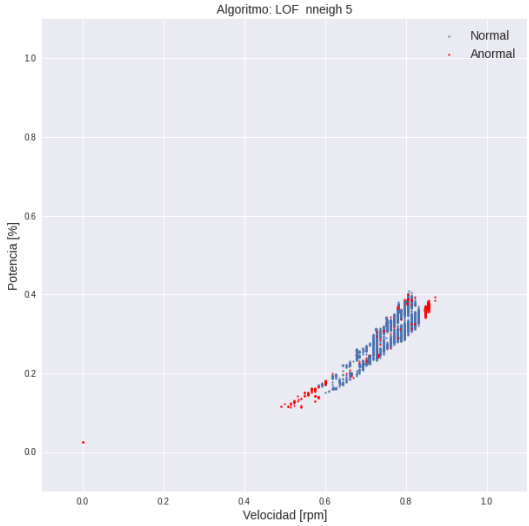


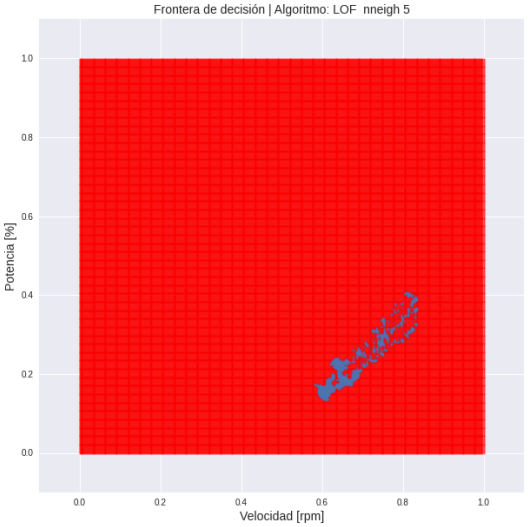


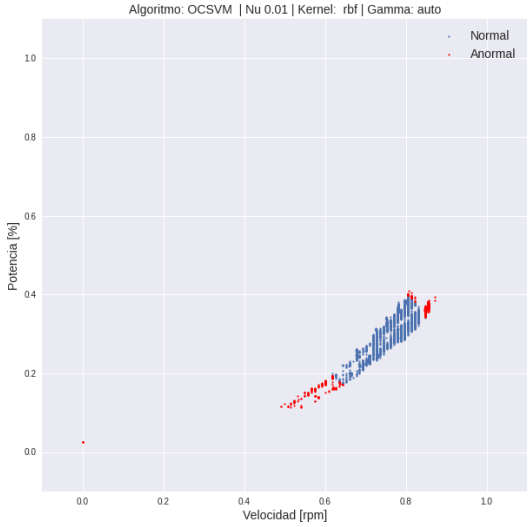


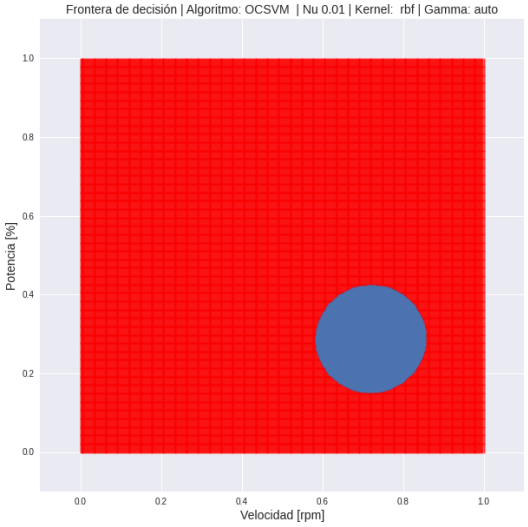

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

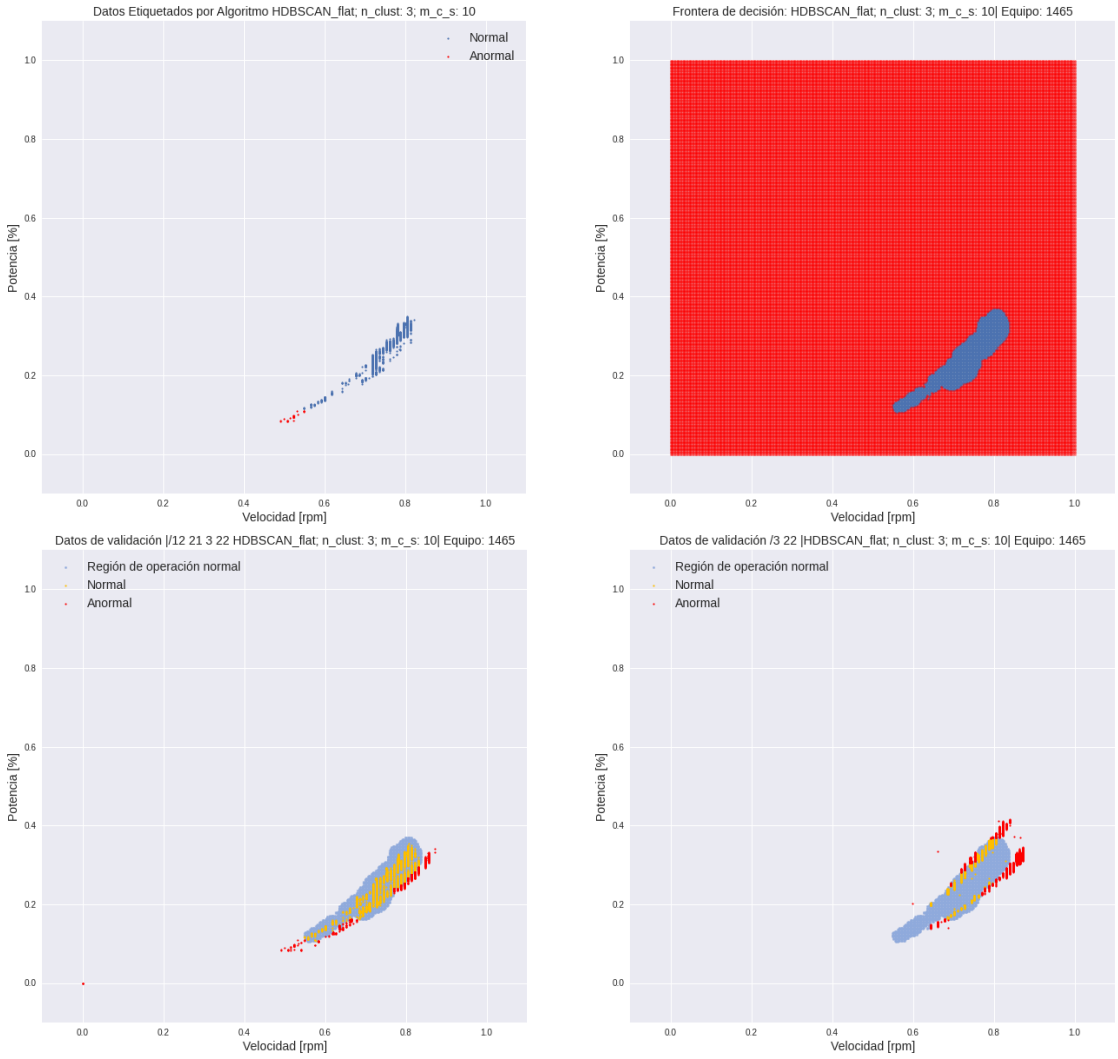

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

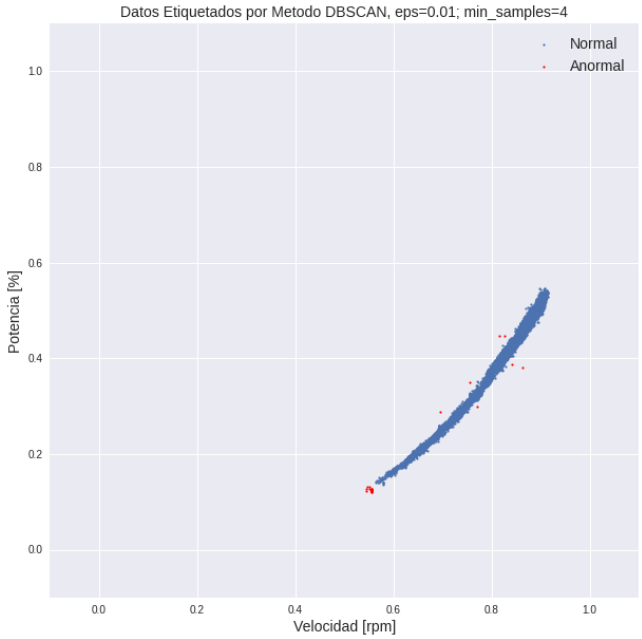

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

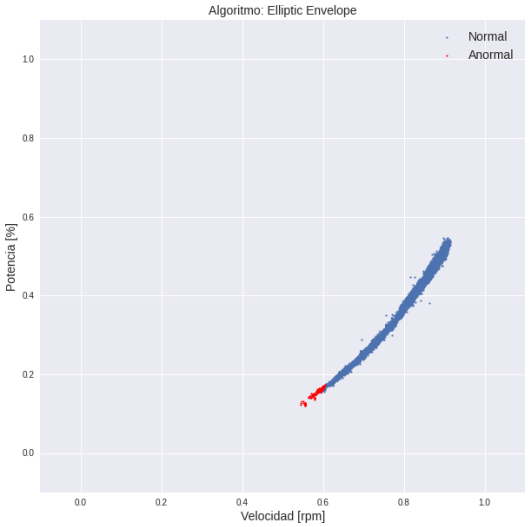

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

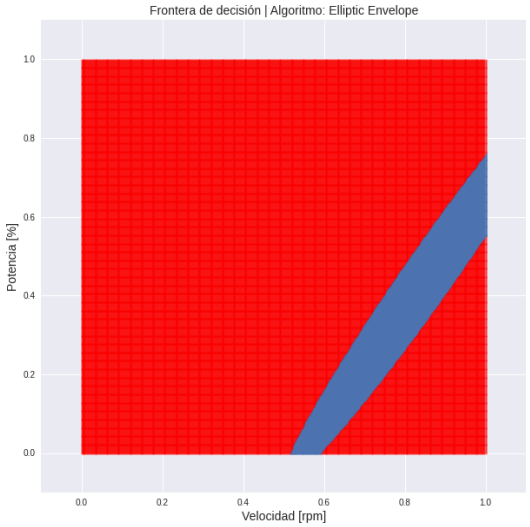
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9384	0.6505	0.2506	0.3618
Isolation Forest	0.9399	0.6881	0.2531	0.3701
Local Outlier Factor	0.9924	0.9270	0.9663	0.9463
OC-SVM	0.9905	0.8811	0.9975	0.9357

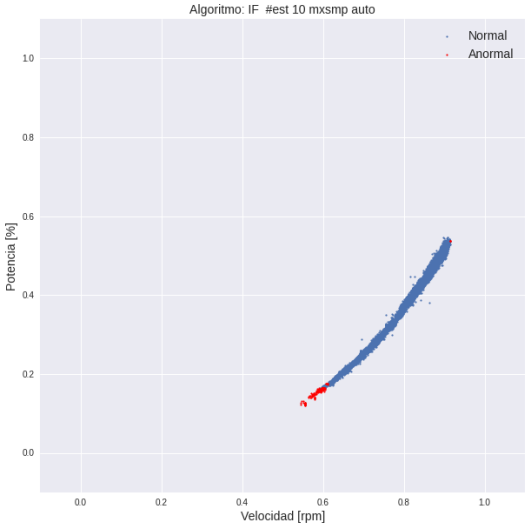


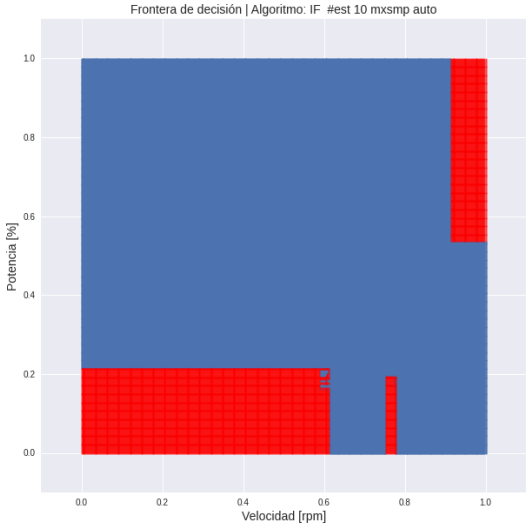
Extractor 2464

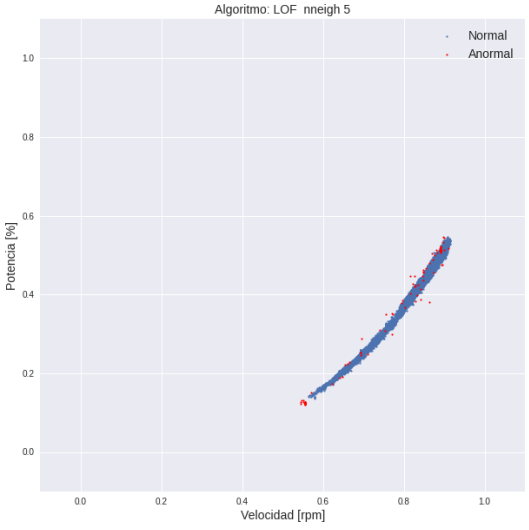


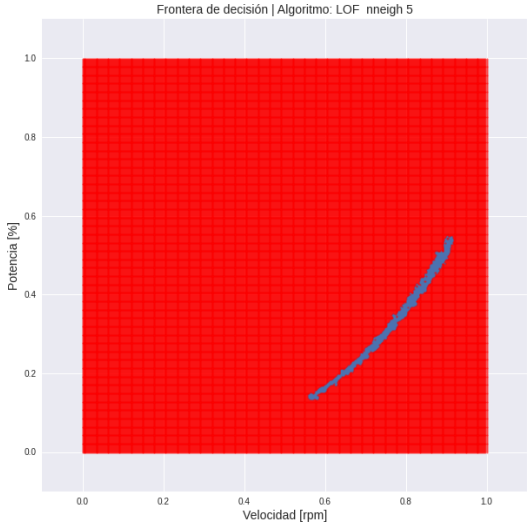


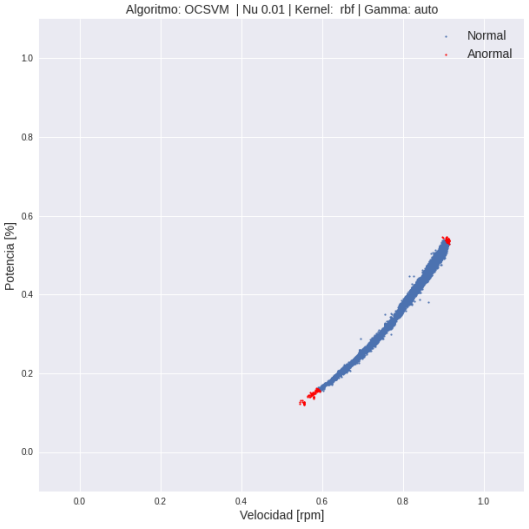


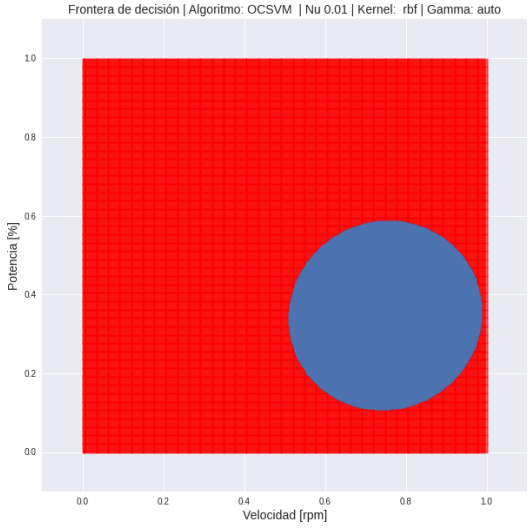

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

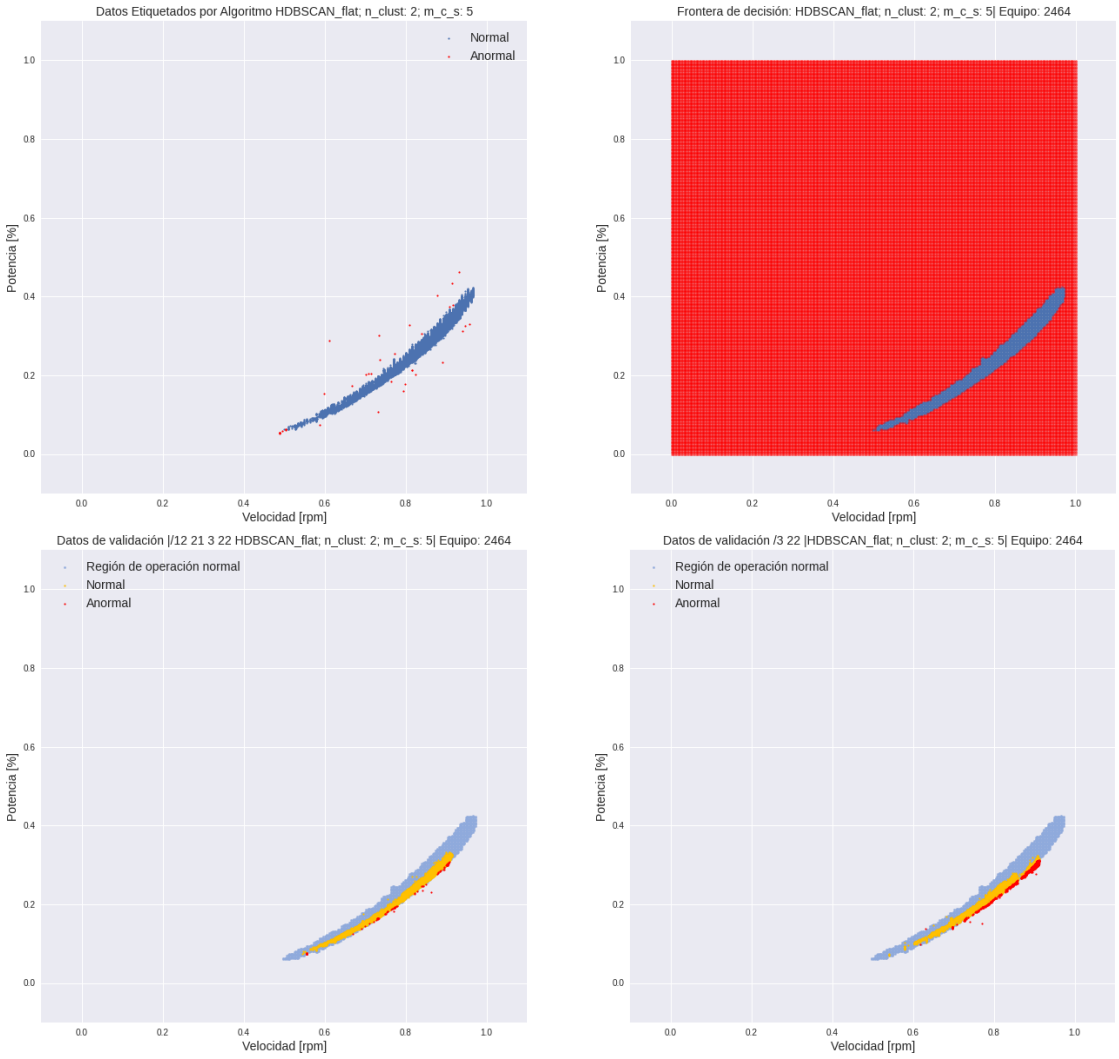

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

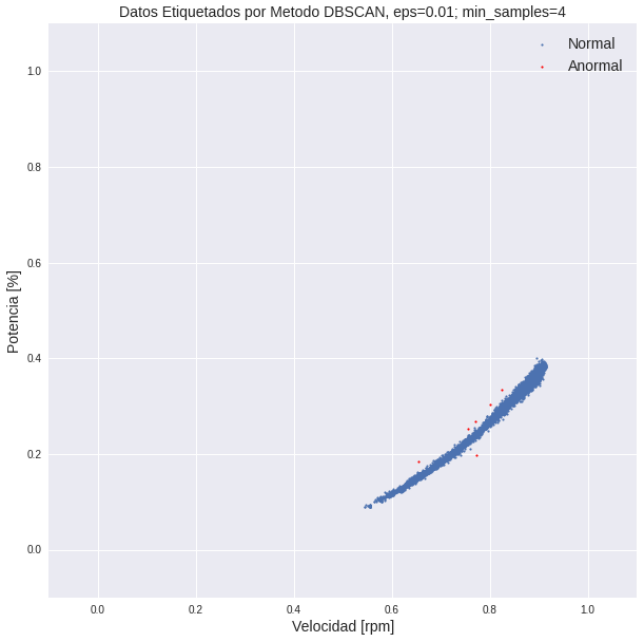

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

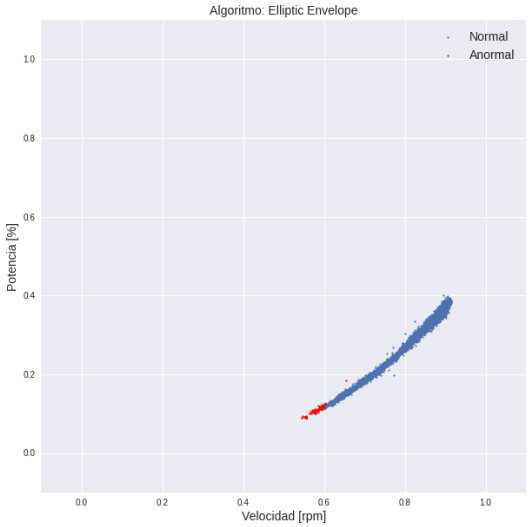

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

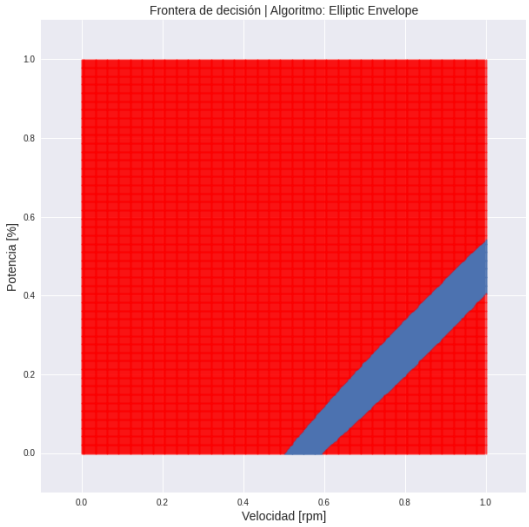
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9894	0.2053	0.8158	0.3280
Isolation Forest	0.9894	0.2053	0.8158	0.3280
Local Outlier Factor	0.9949	0.3878	1.0000	0.5588
OC-SVM	0.9896	0.2081	0.8158	0.3316

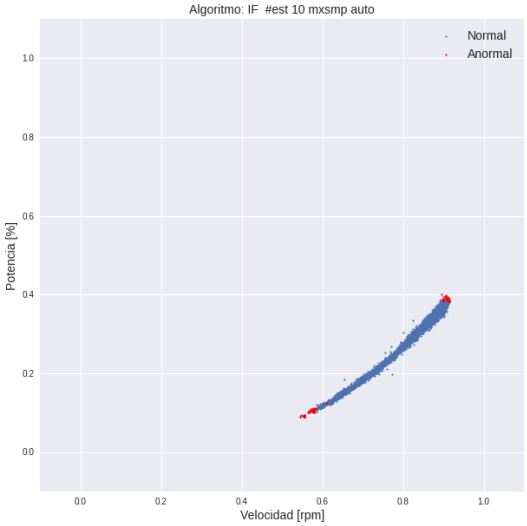


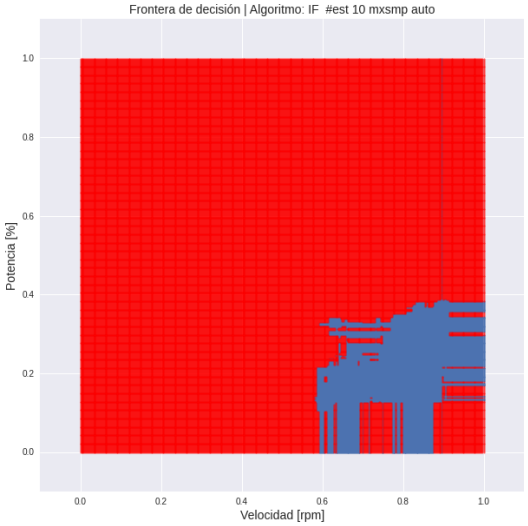
Extractor 2465

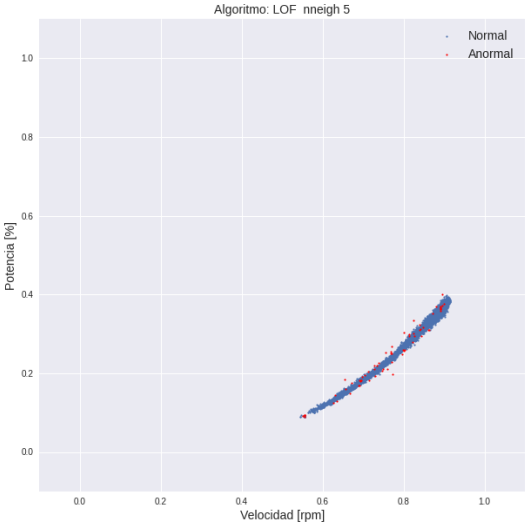


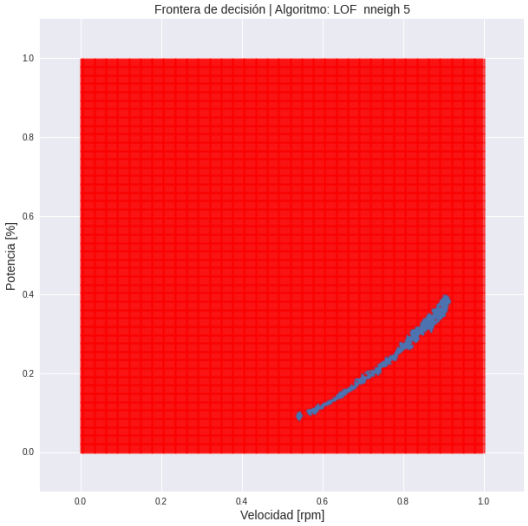


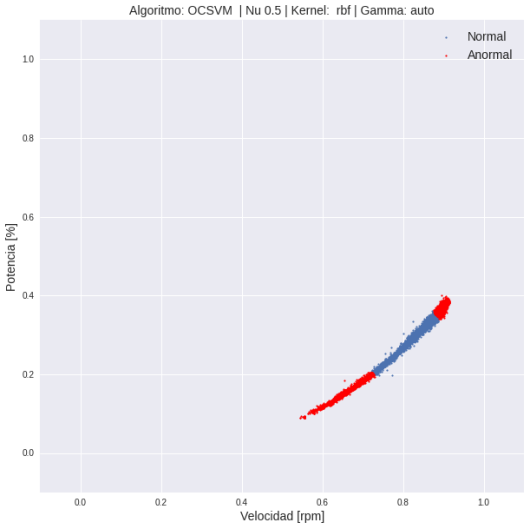


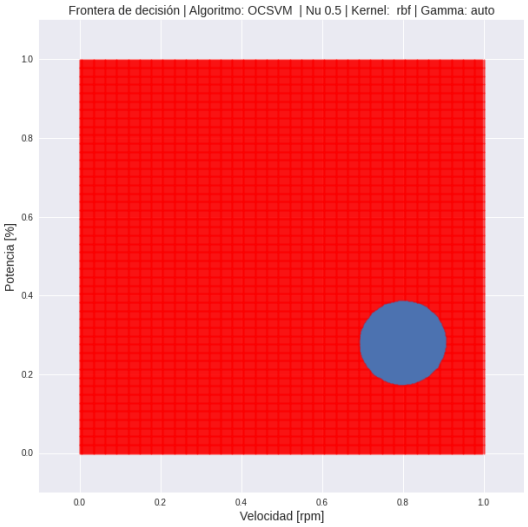

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

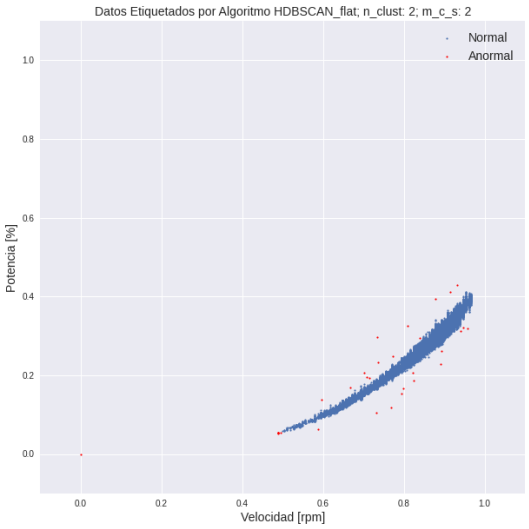

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

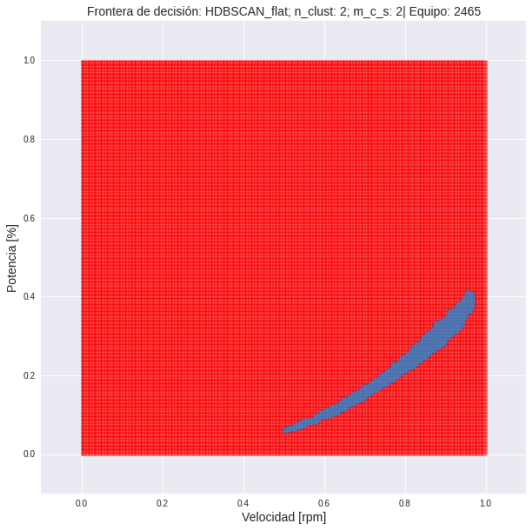

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

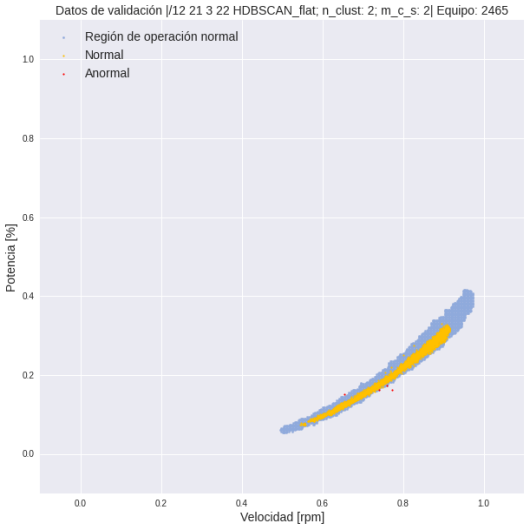

Algoritmo: OCSVM | Nu 0.5 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

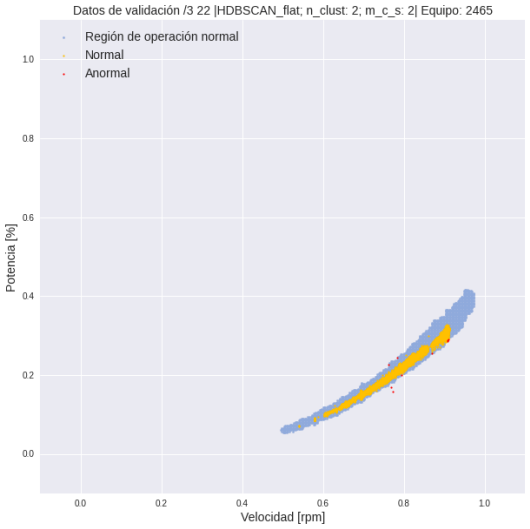

Frontera de decisión | Algoritmo: OCSVM | Nu 0.5 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9896	0.0083	0.1667	0.0157
Isolation Forest	0.9901	0.0000	0.0000	0.0000
Local Outlier Factor	0.9949	0.0909	1.0000	0.1667
OC-SVM	0.4998	0.0002	0.1667	0.0003

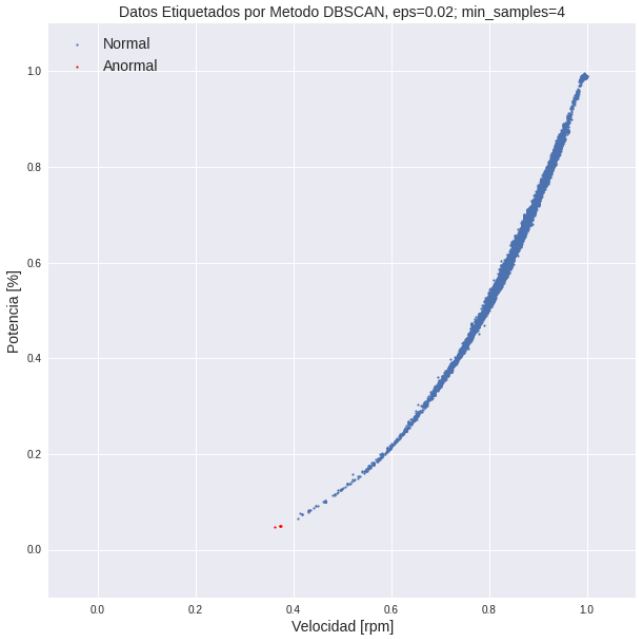


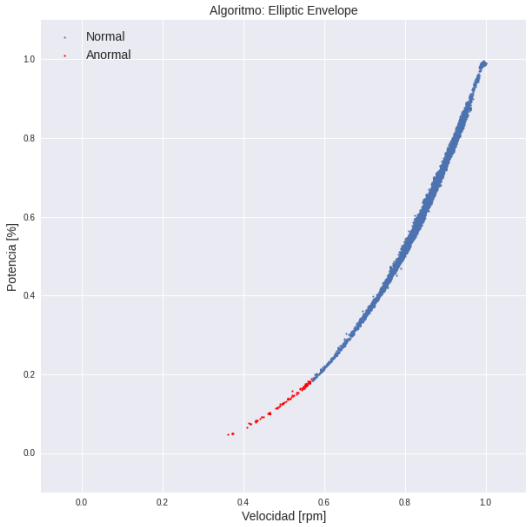


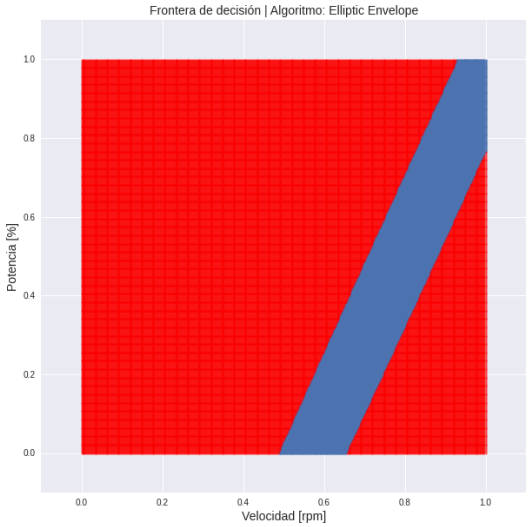


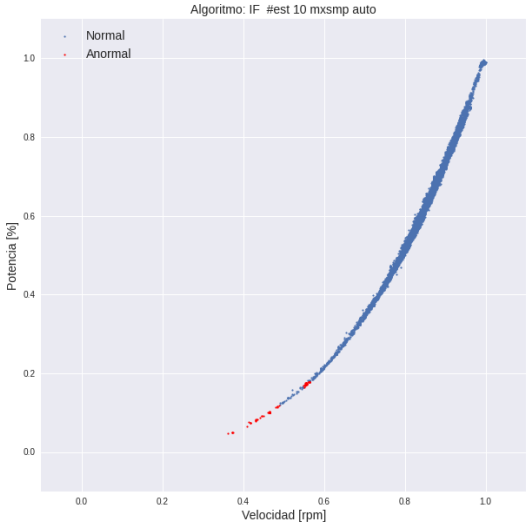


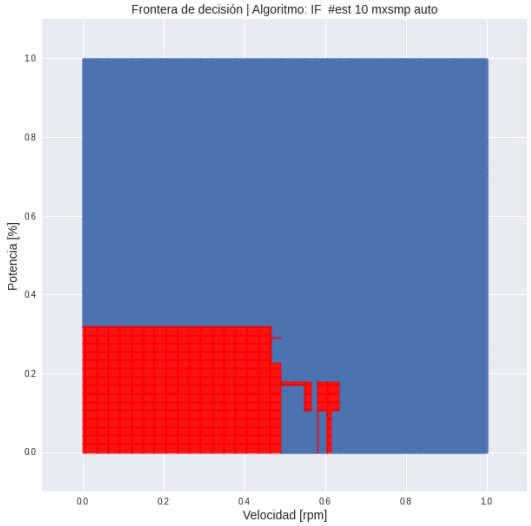
Inductor 1461

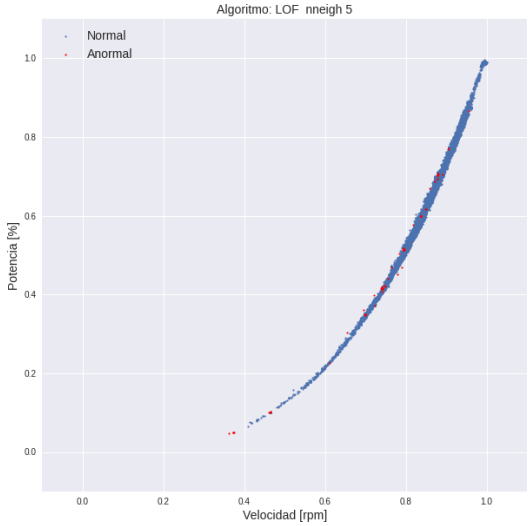


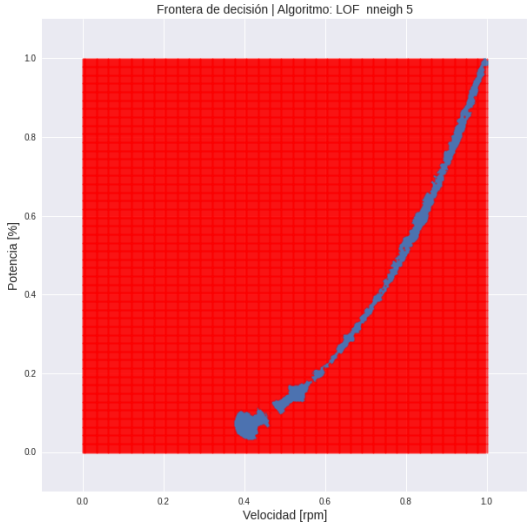


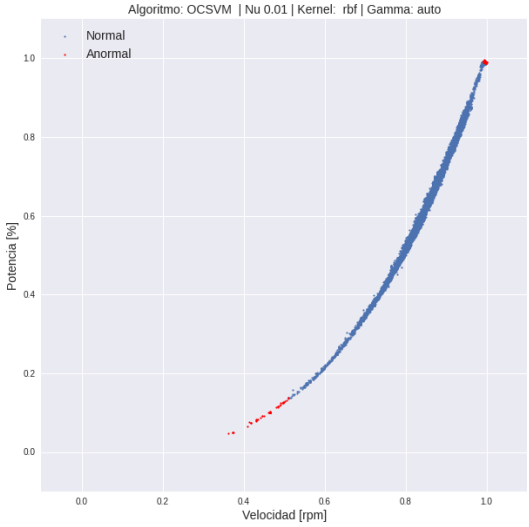


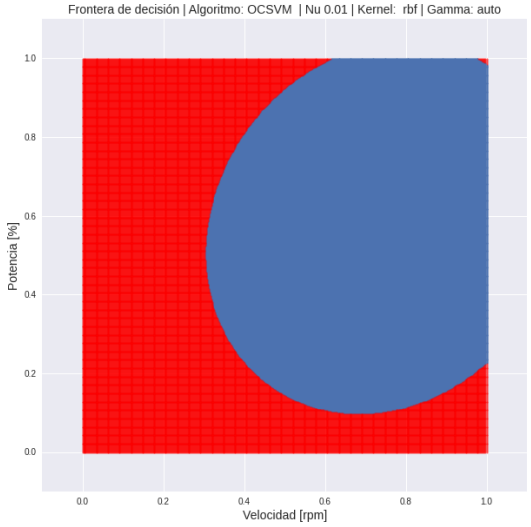

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

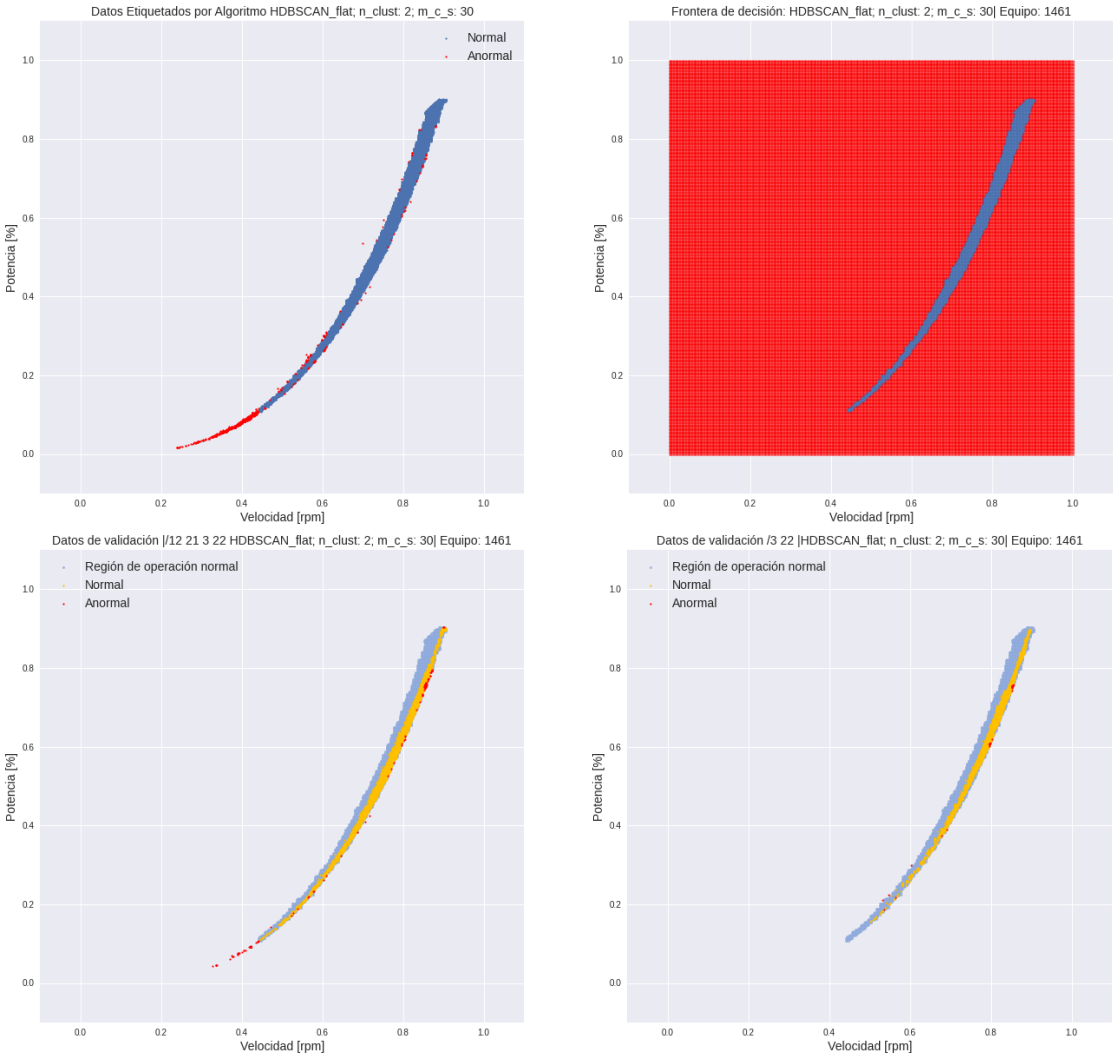

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

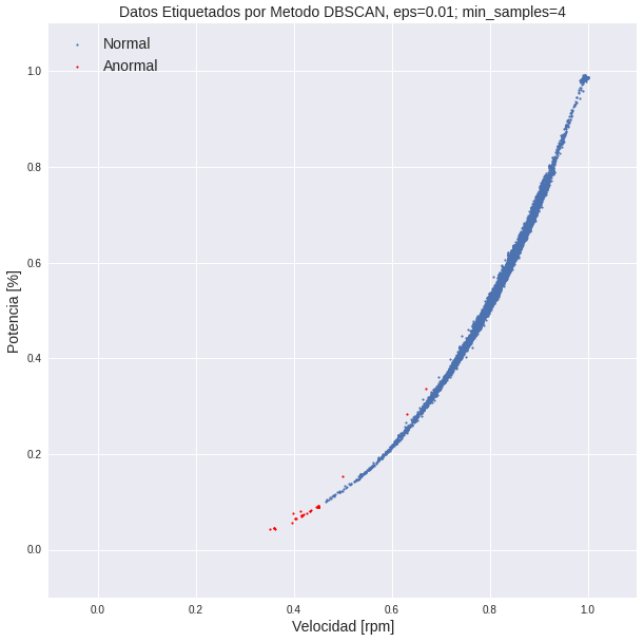

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

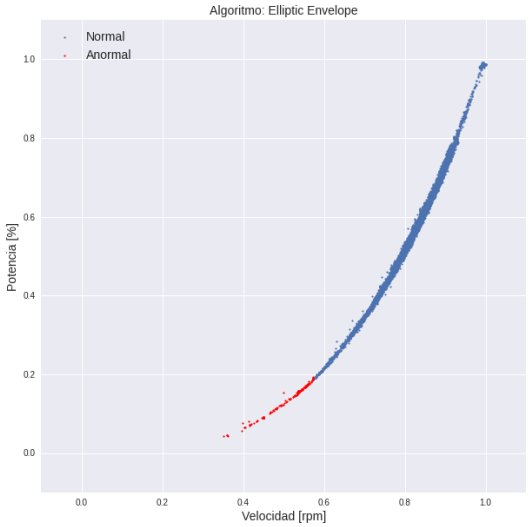

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

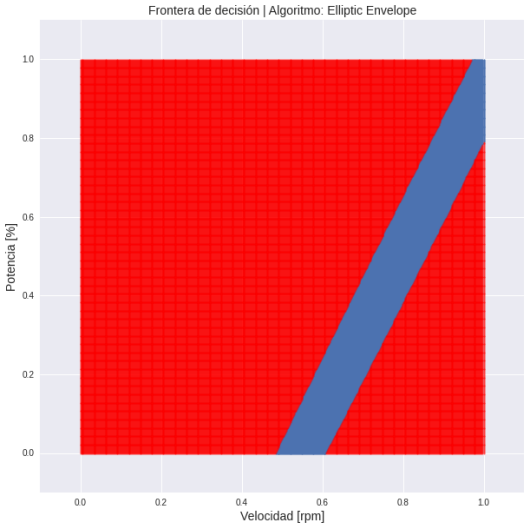
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0413	1.0000	0.0794
Isolation Forest	0.9937	0.0649	1.0000	0.1219
Local Outlier Factor	0.9946	0.0746	1.0000	0.1389
OC-SVM	0.9899	0.0413	1.0000	0.0794

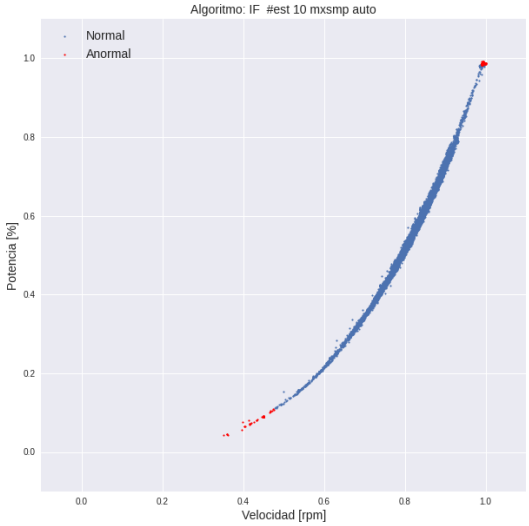


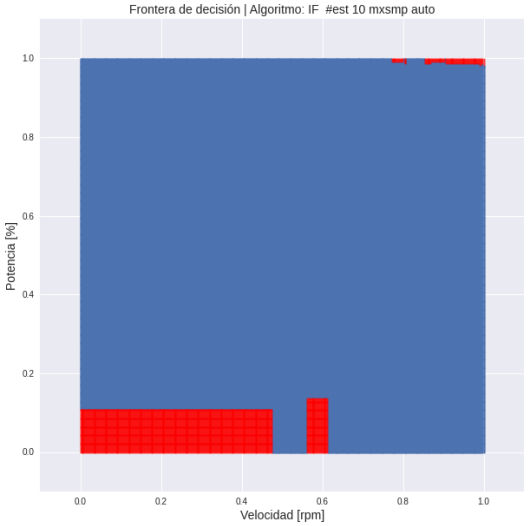
Inductor 1462

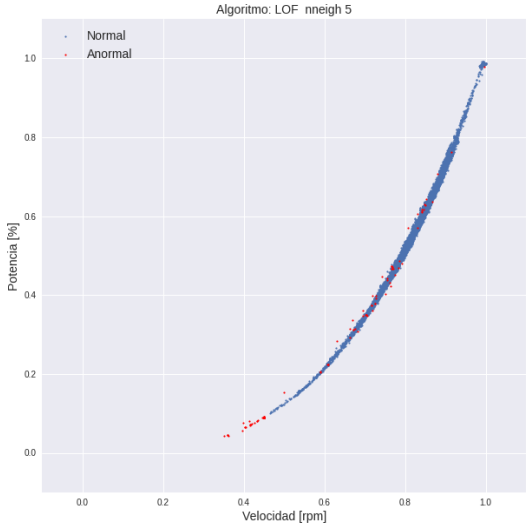


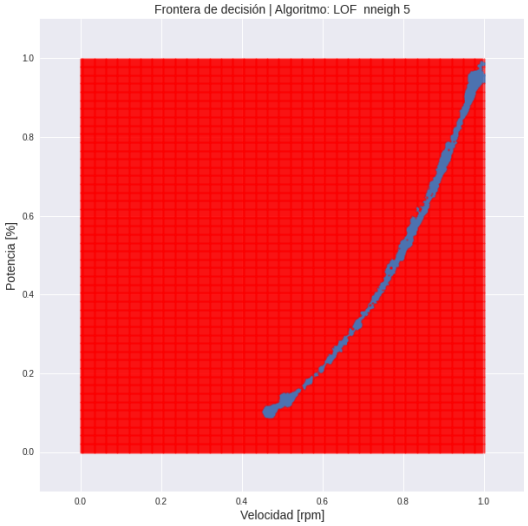


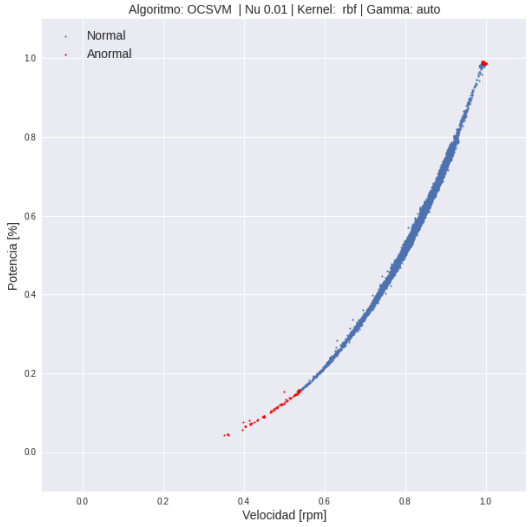


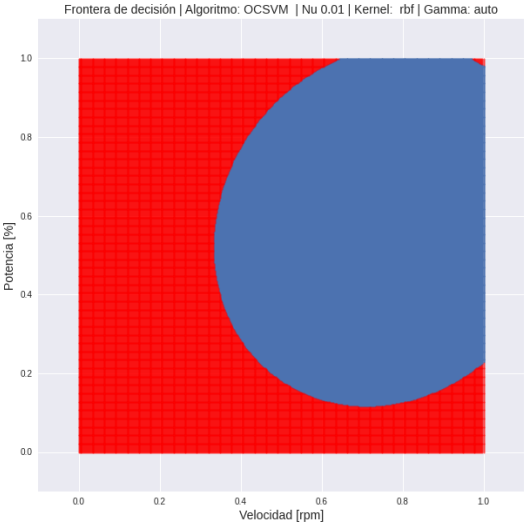

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

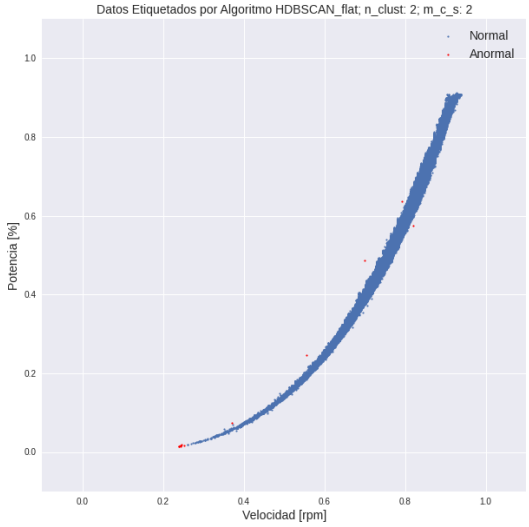

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

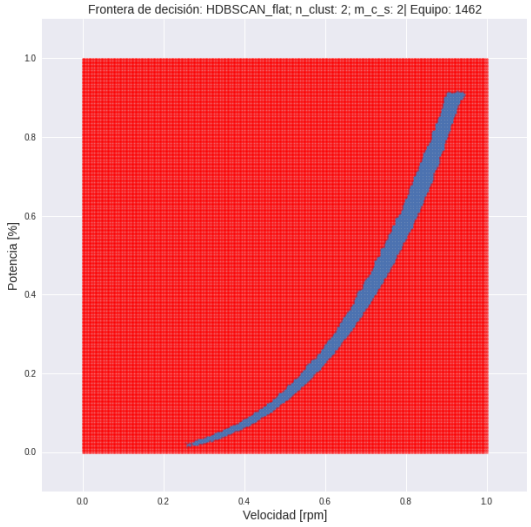

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

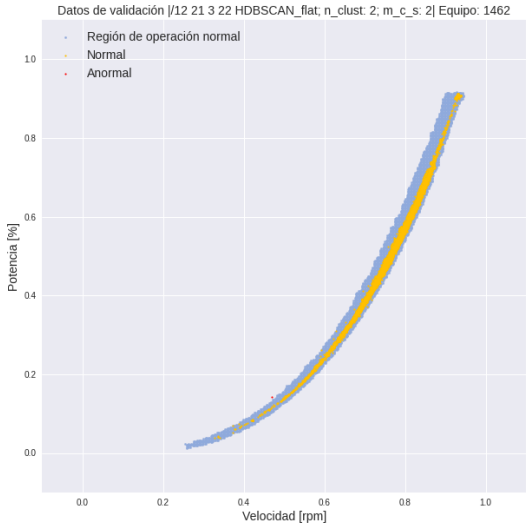

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

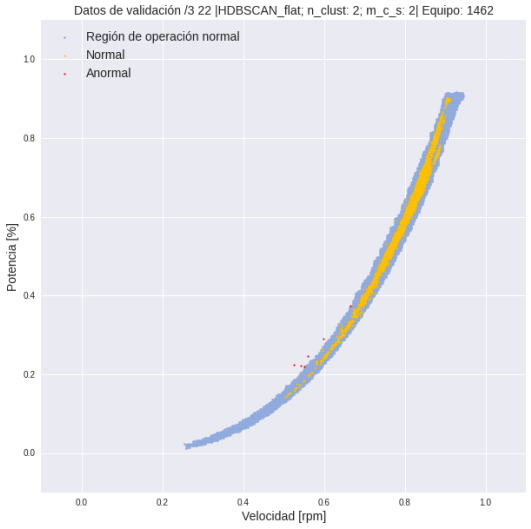

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9898	0.2857	0.9583	0.4402
Isolation Forest	0.9927	0.3571	0.9375	0.5172
Local Outlier Factor	0.9952	0.4660	1.0000	0.6358
OC-SVM	0.9900	0.2893	0.9583	0.4444

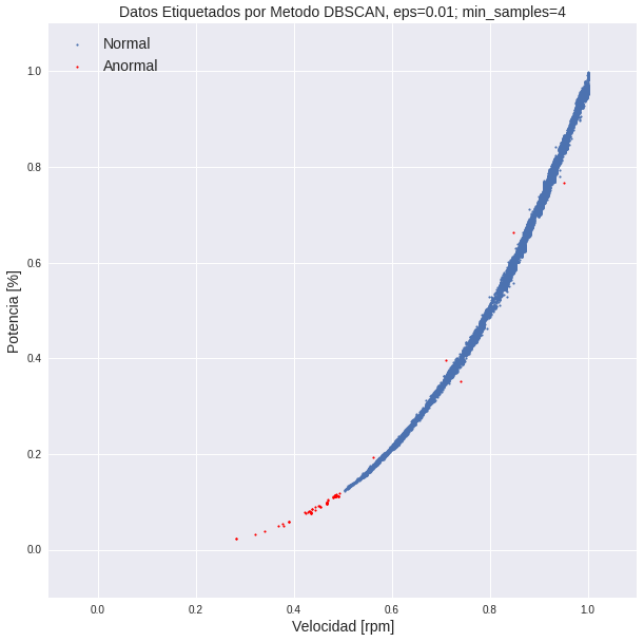


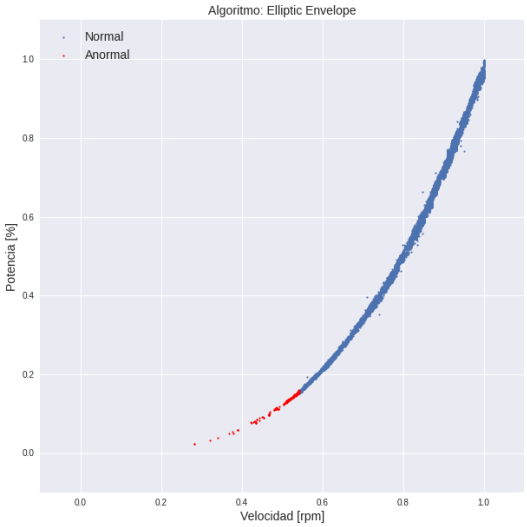


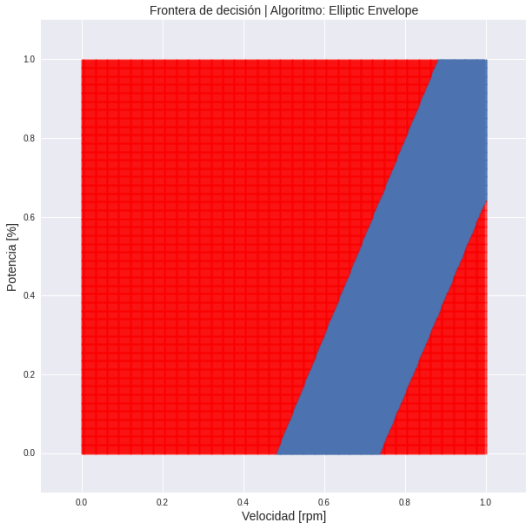


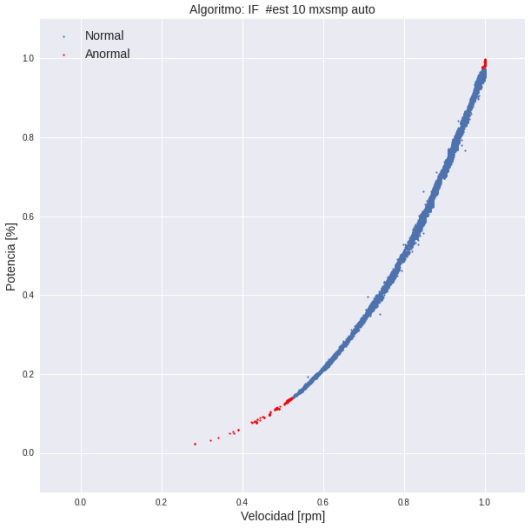


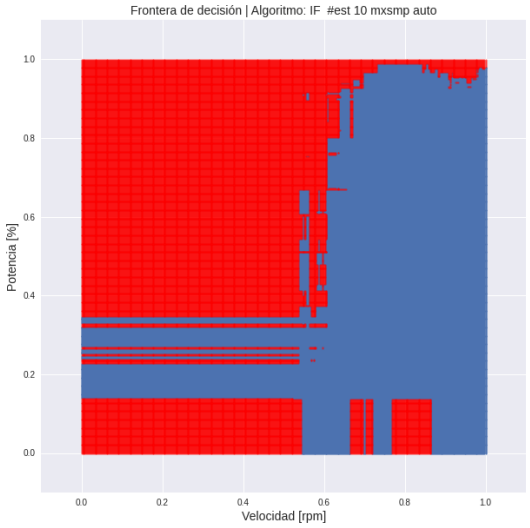
Inductor 2461

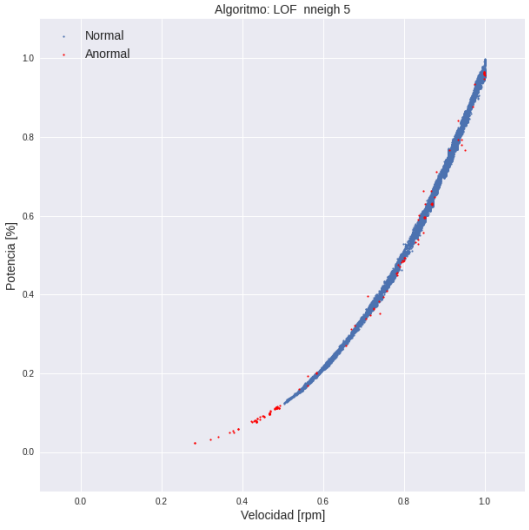


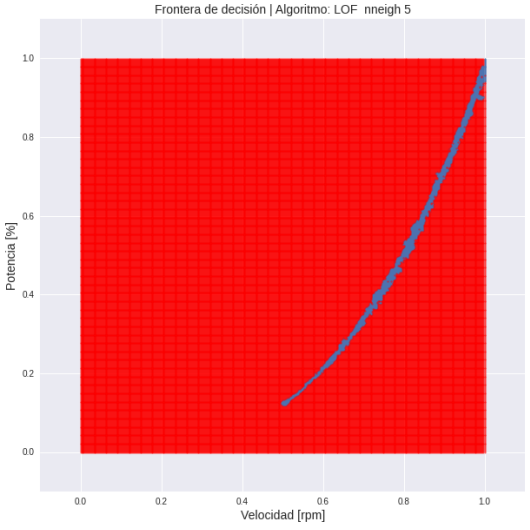


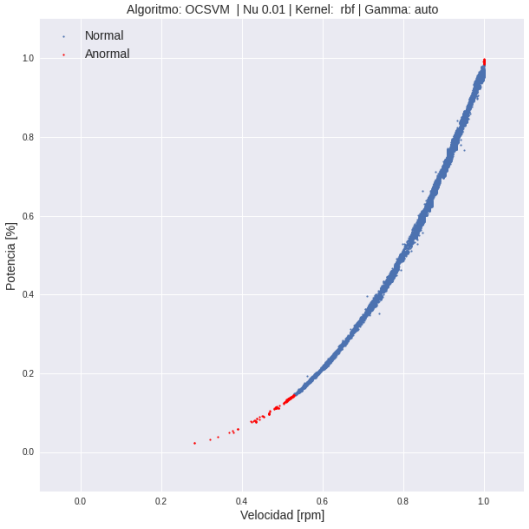


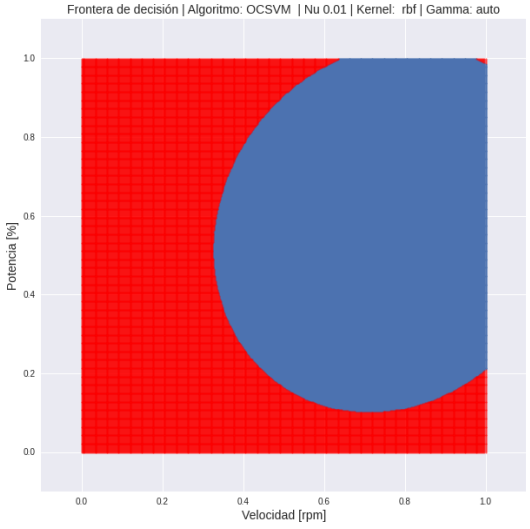

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

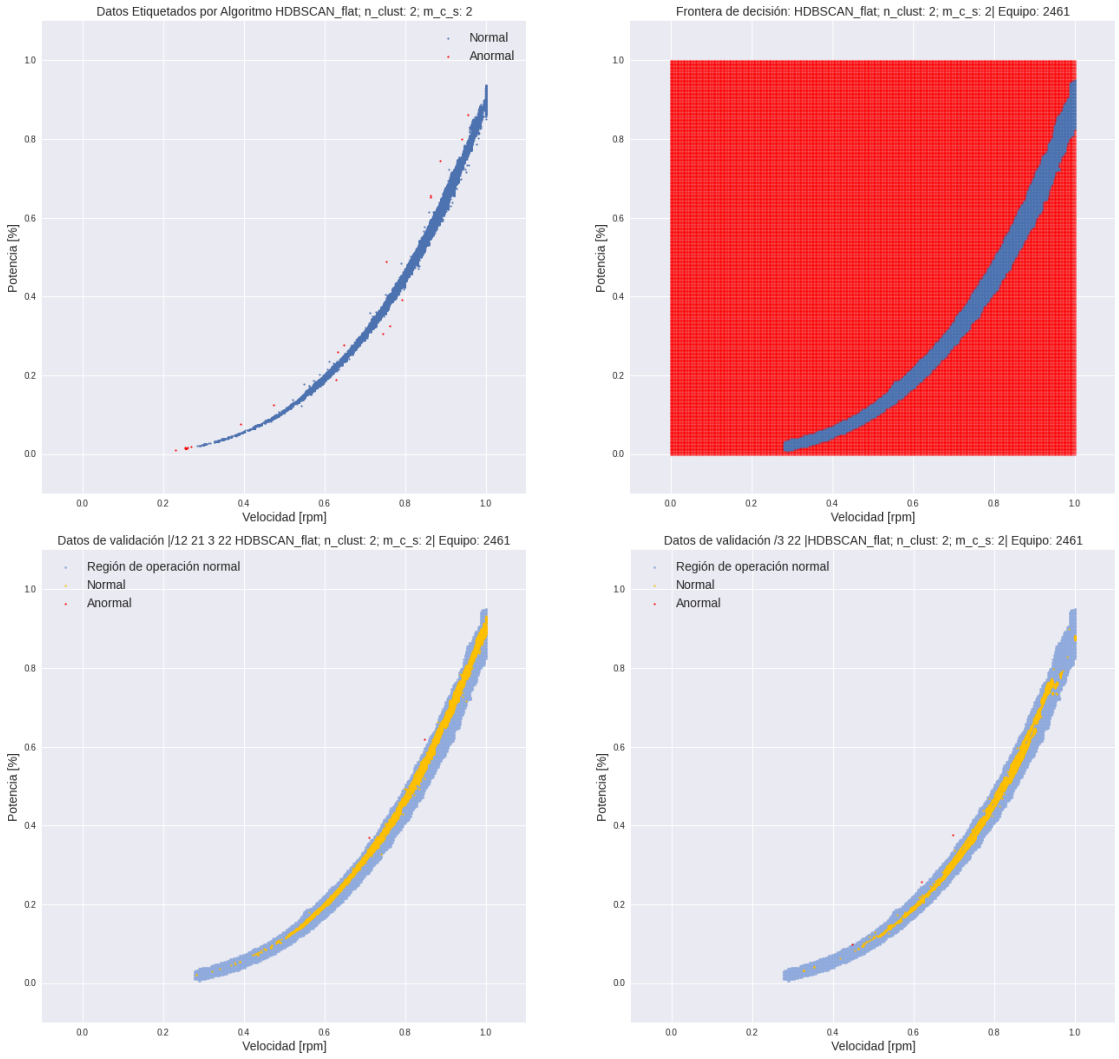

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

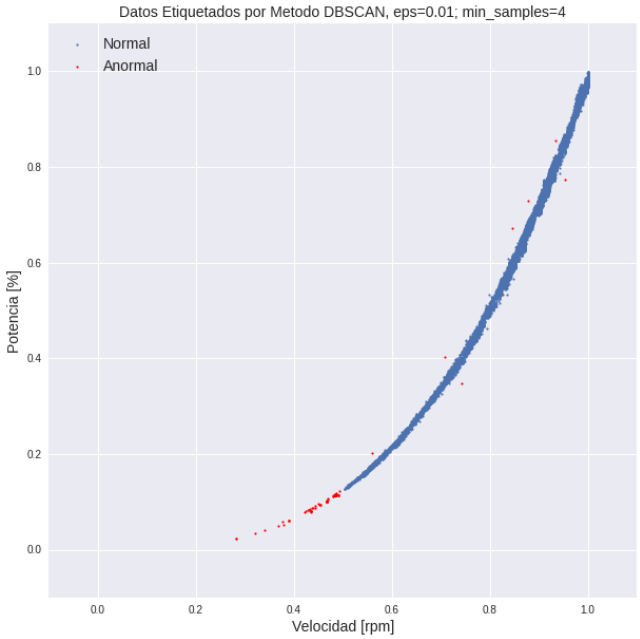

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

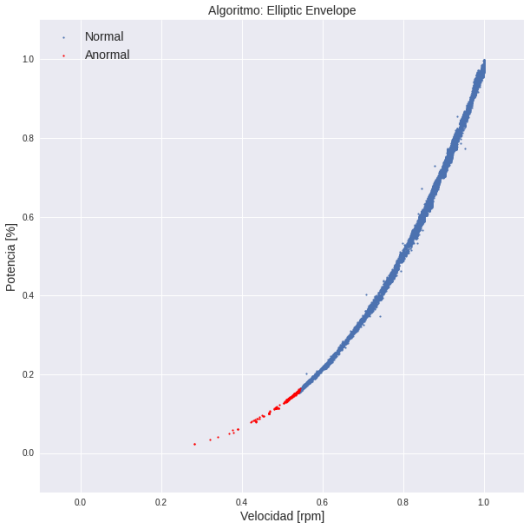

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

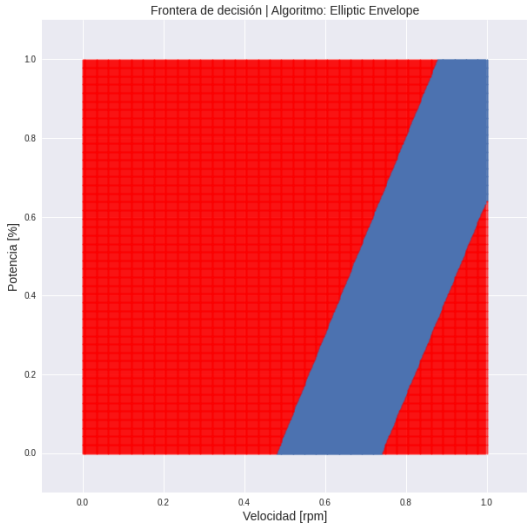
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9896	0.3878	0.9383	0.5487
Isolation Forest	0.9896	0.3878	0.9383	0.5474
Local Outlier Factor	0.9949	0.5704	1.0000	0.7265
OC-SVM	0.9897	0.3918	0.9383	0.5527

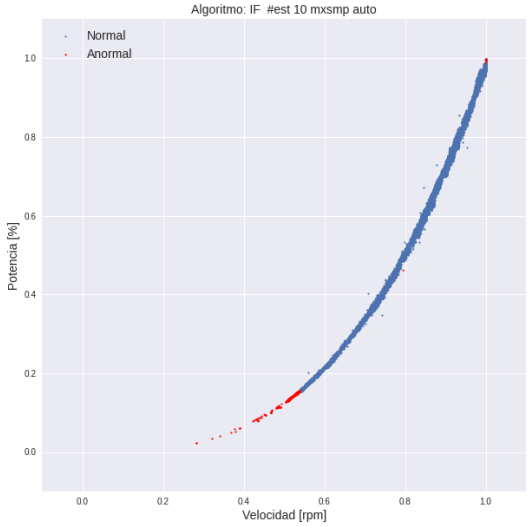


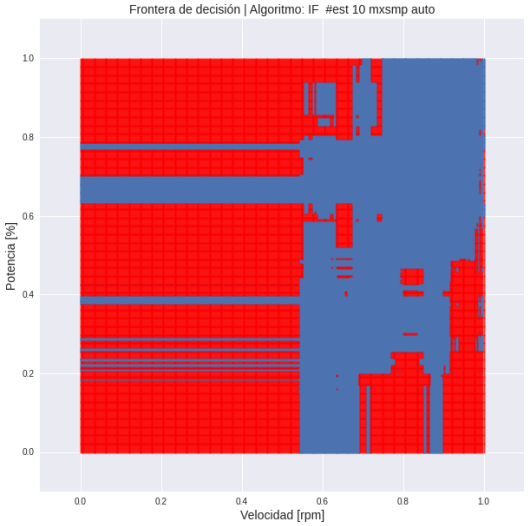
Inductor 2462

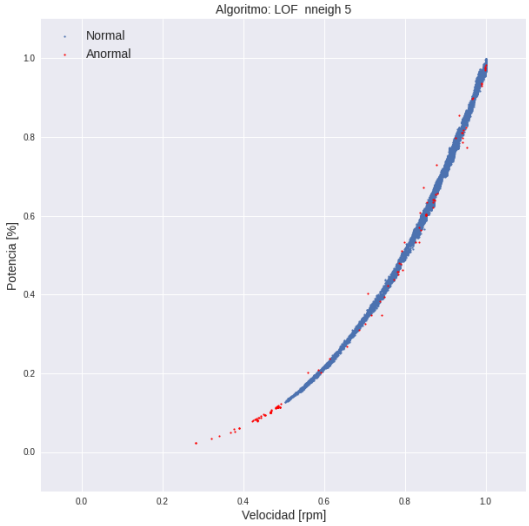


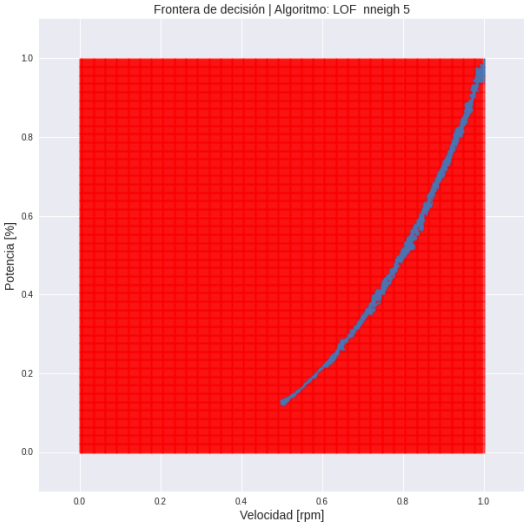


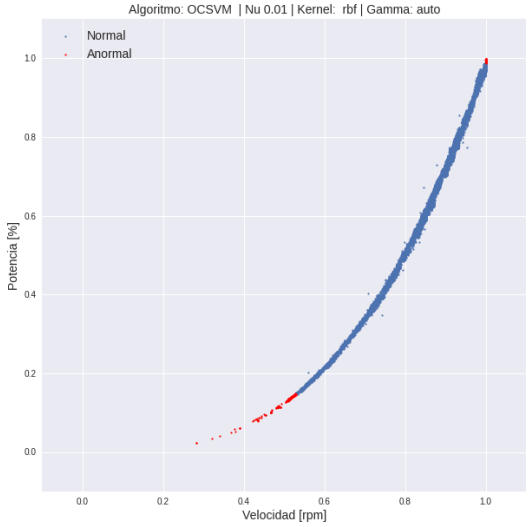


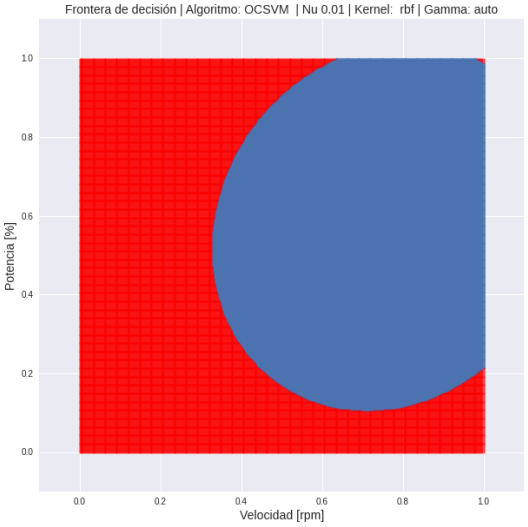

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

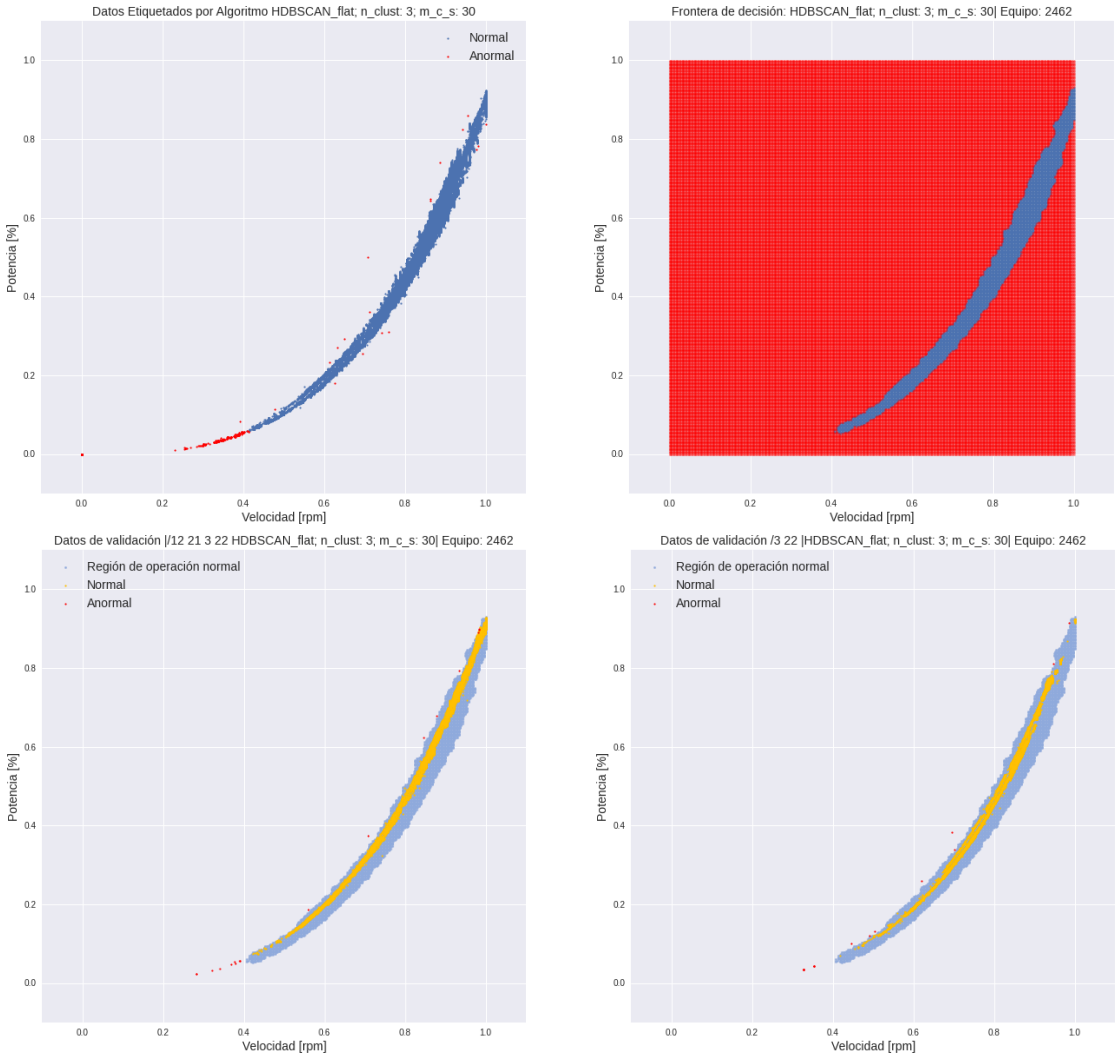

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

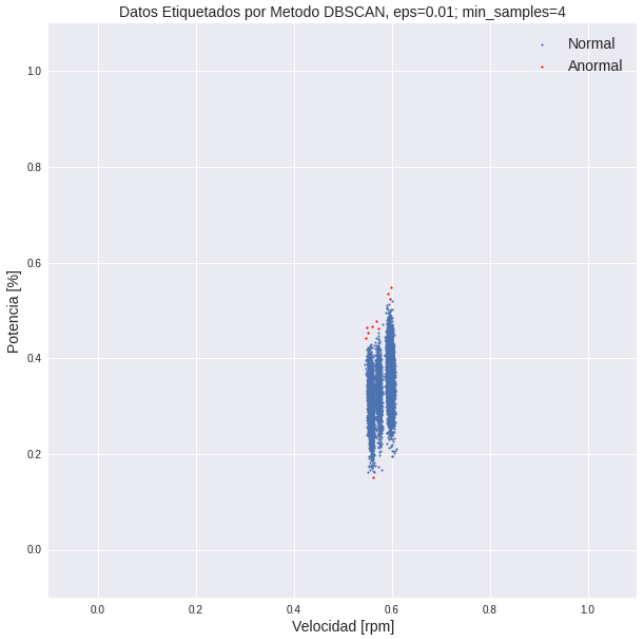

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

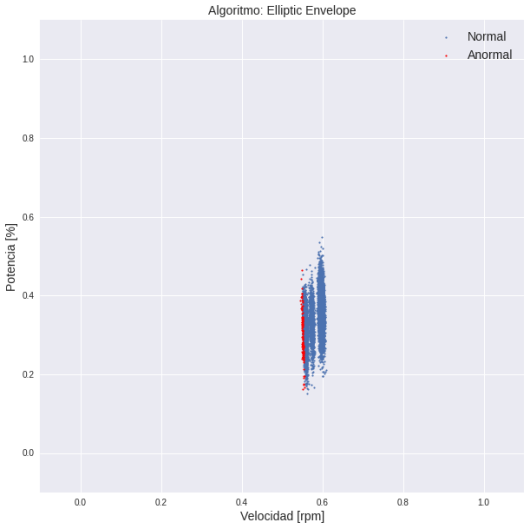

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

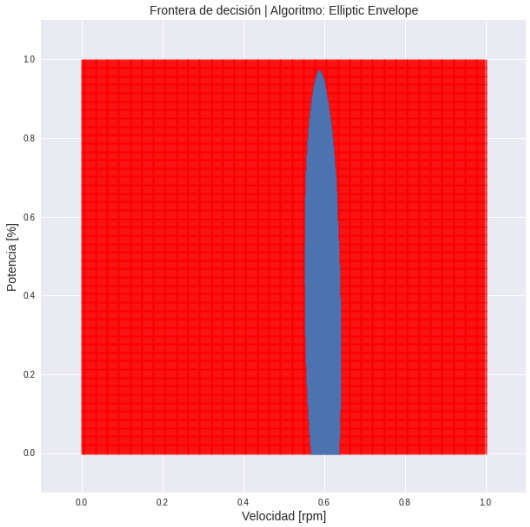
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9894	0.3878	0.9157	0.5448
Isolation Forest	0.9907	0.4199	0.9157	0.5758
Local Outlier Factor	0.9952	0.5929	1.0000	0.7444
OC-SVM	0.9896	0.3918	0.9157	0.5487

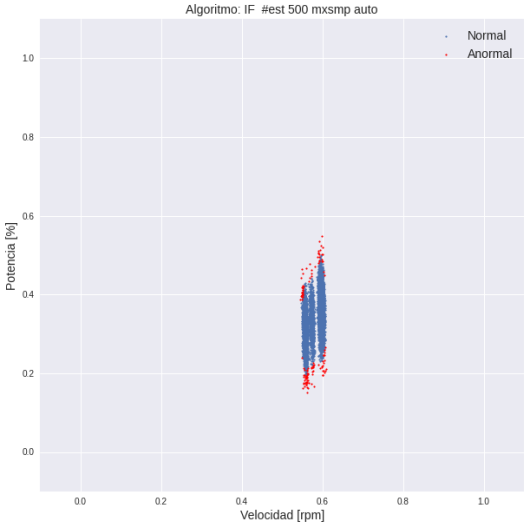


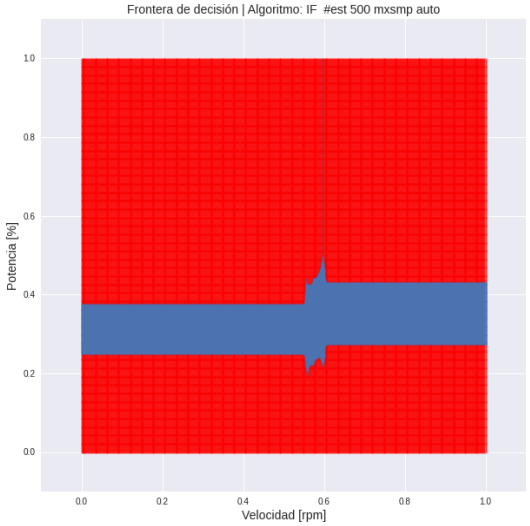
Soplador 1355

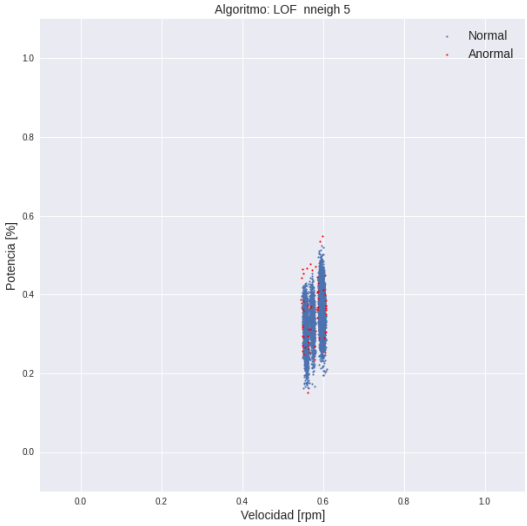


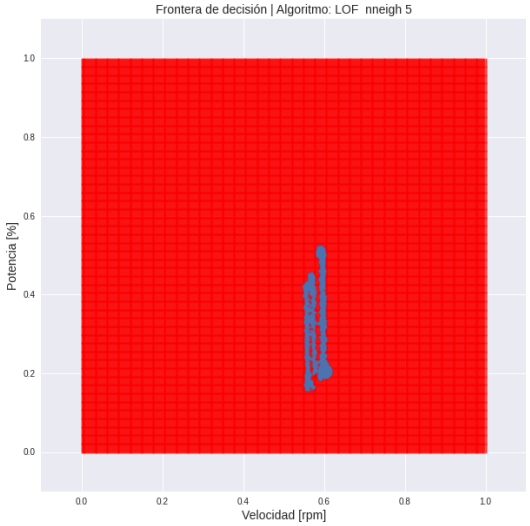


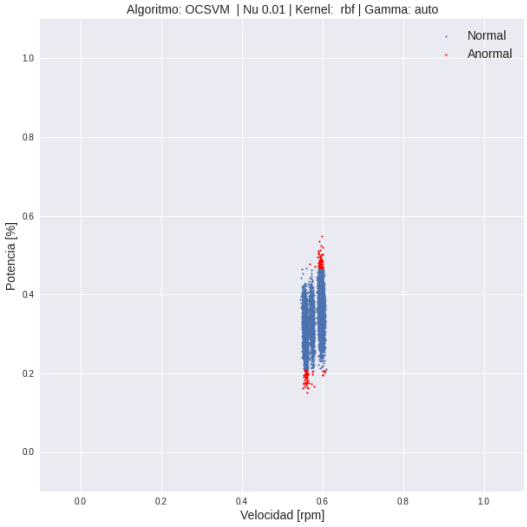


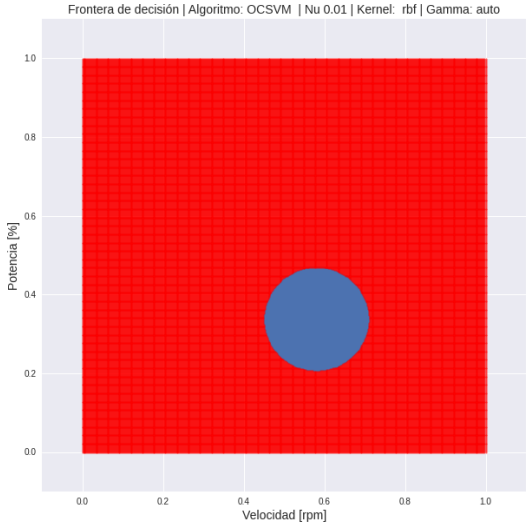

Algoritmo: IF #est 500 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 500 mxsmp auto
Potencia [%]
Velocidad [rpm]

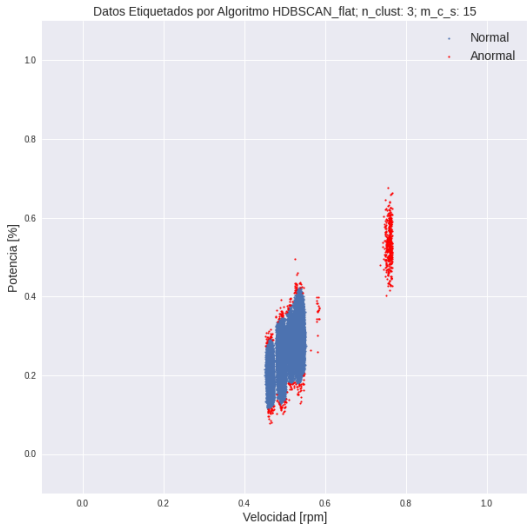

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

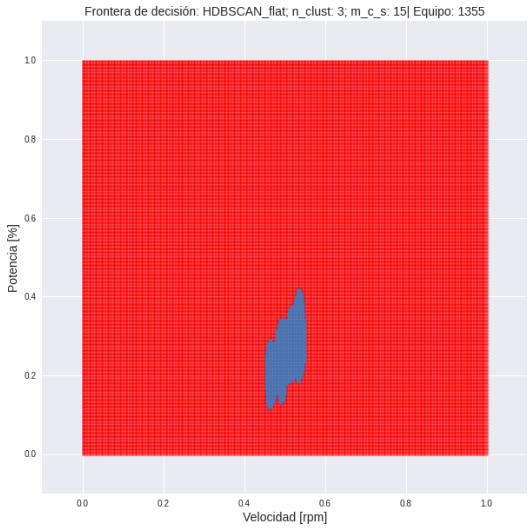

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

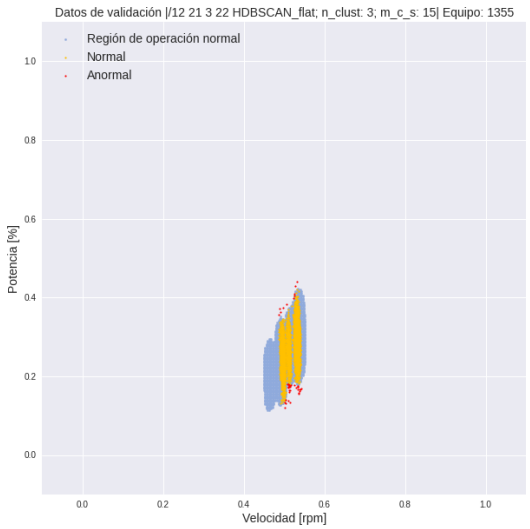

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

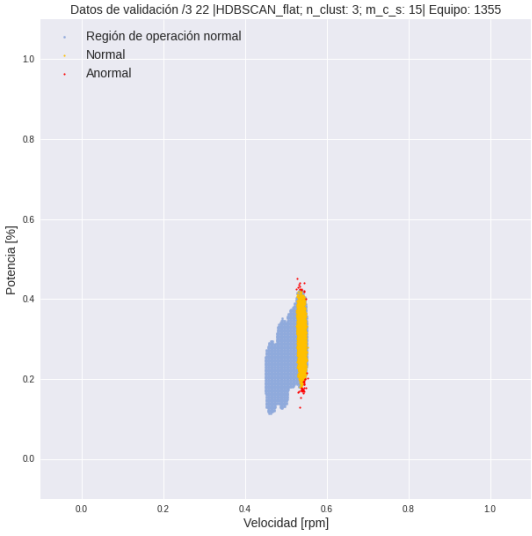

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9892	0.0169	0.2000	0.0313
Isolation Forest	0.9899	0.0794	1.0000	0.1471
Local Outlier Factor	0.9954	0.1475	0.9000	0.2535
OC-SVM	0.9897	0.0420	0.5000	0.0775

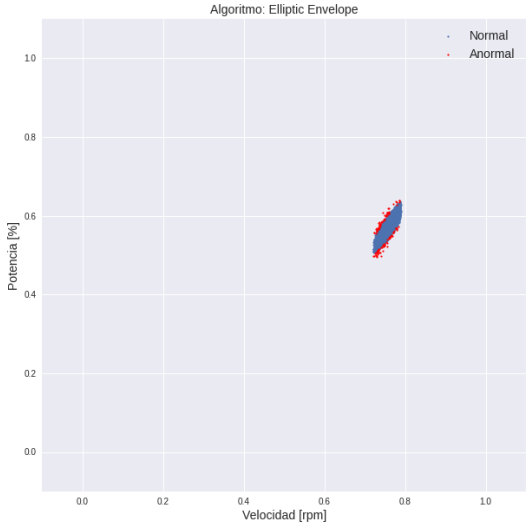


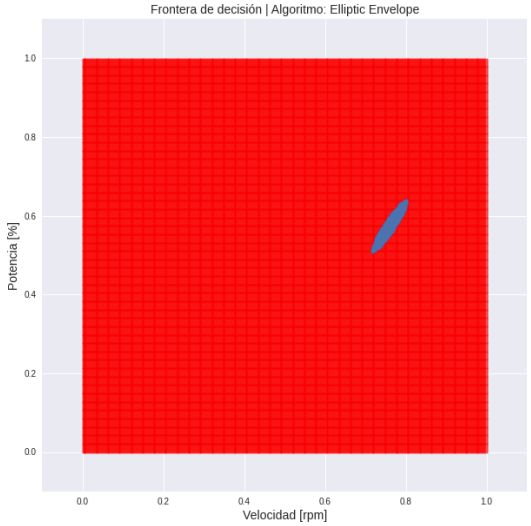


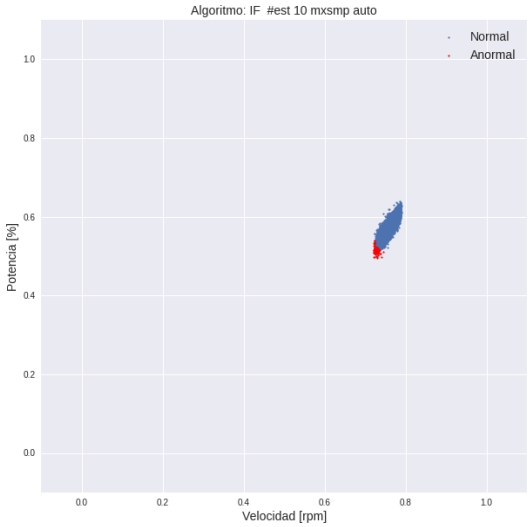


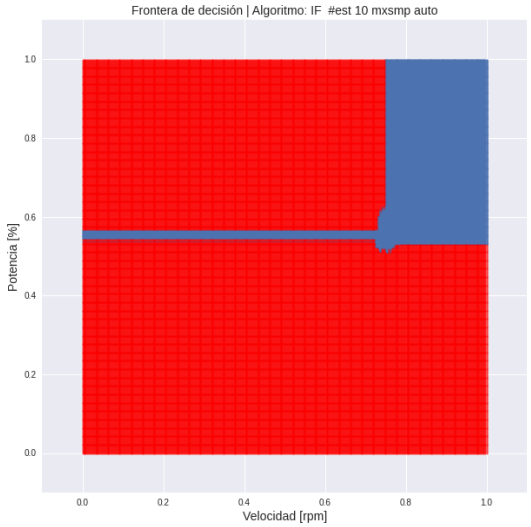


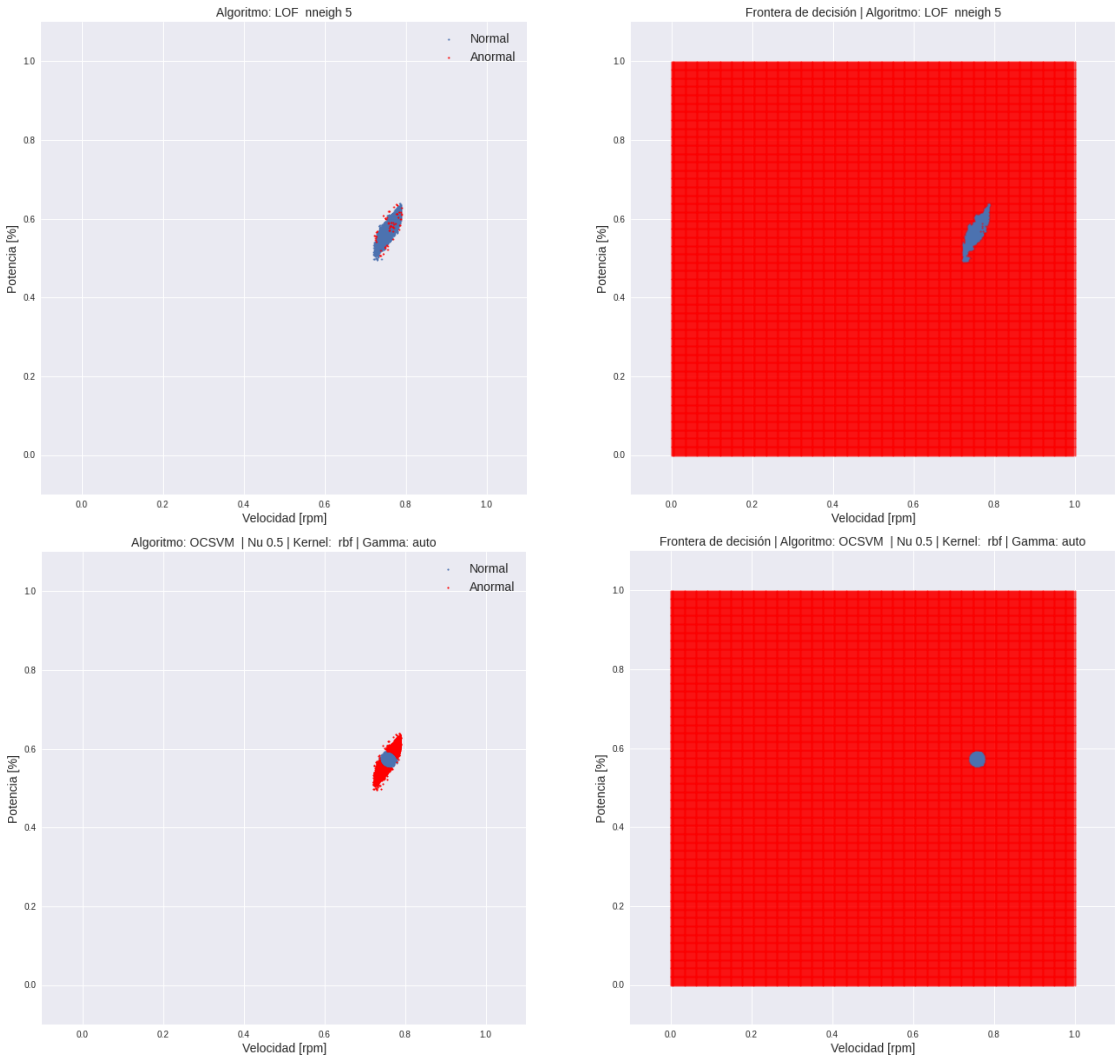
Soplador 2355



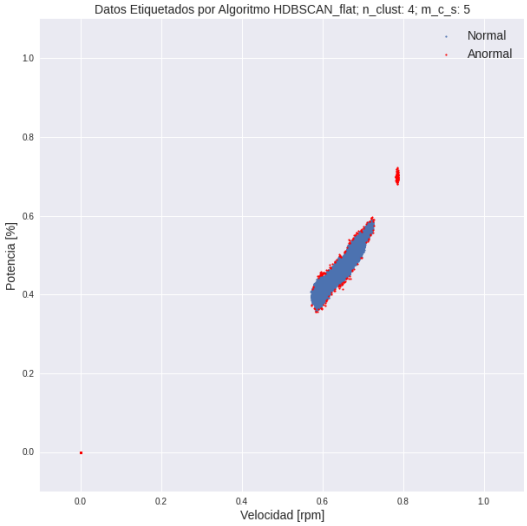


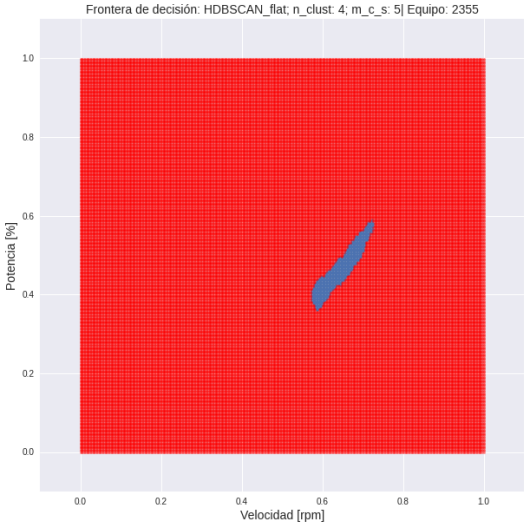


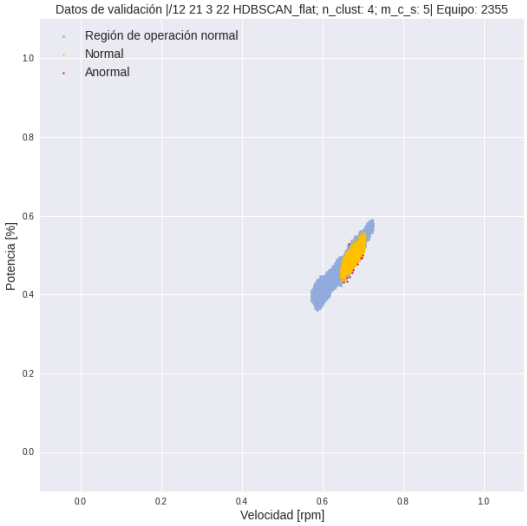


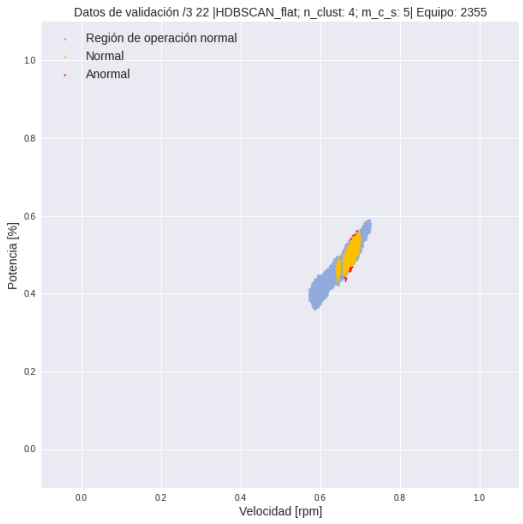


Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9899	0.0000	0.0000	0.0000
Isolation Forest	0.9902	0.0000	0.0000	0.0000
Local Outlier Factor	0.9953	0.0000	0.0000	0.0000
OC-SVM	0.5000	0.0000	0.0000	0.0000

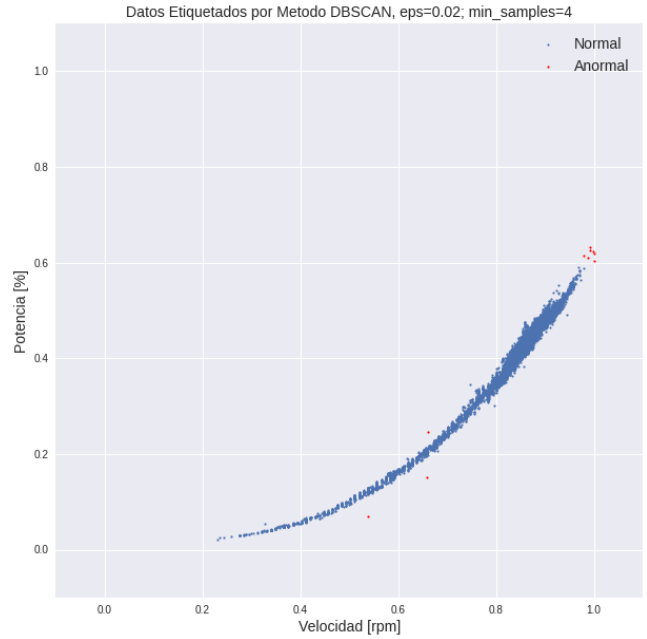

Datos Etiquetados por Algoritmo HDBSCAN_flat; n_clust: 4; m_c_s: 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

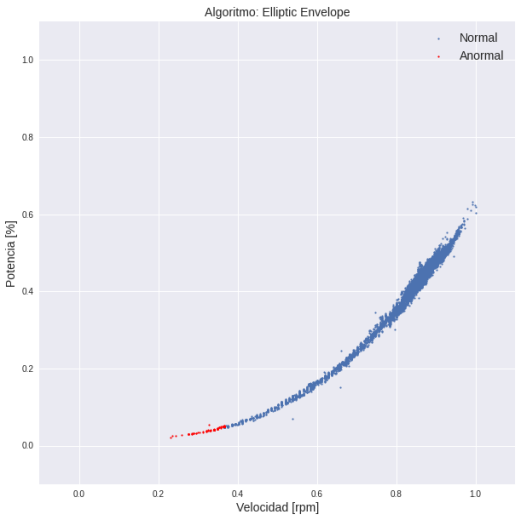

Frontera de decisión: HDBSCAN_flat; n_clust: 4; m_c_s: 5| Equipo: 2355
Potencia [%]
Velocidad [rpm]

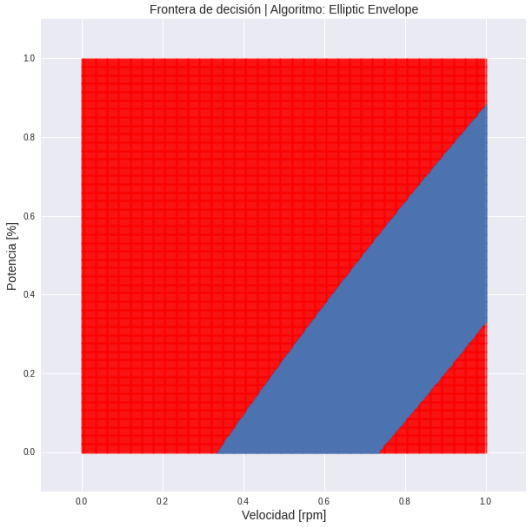

Datos de validación |/12 21 3 22 HDBSCAN_flat; n_clust: 4; m_c_s: 5| Equipo: 2355
Región de operación normal
Normal
Anormal
Potencia [%]
Velocidad [rpm]

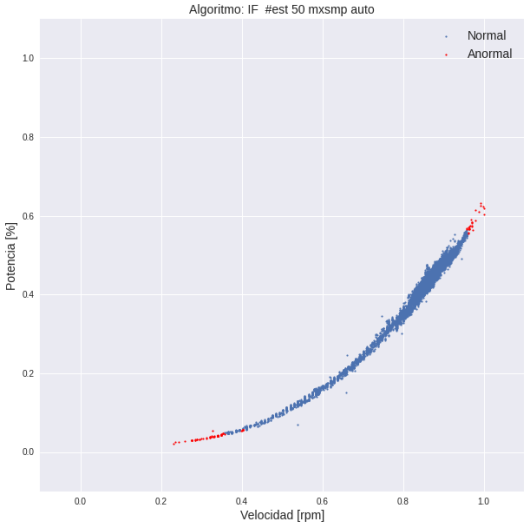

Datos de validación /3 22 |HDBSCAN_flat; n_clust: 4; m_c_s: 5| Equipo: 2355
Región de operación normal
Normal
Anormal
Potencia [%]
Velocidad [rpm]

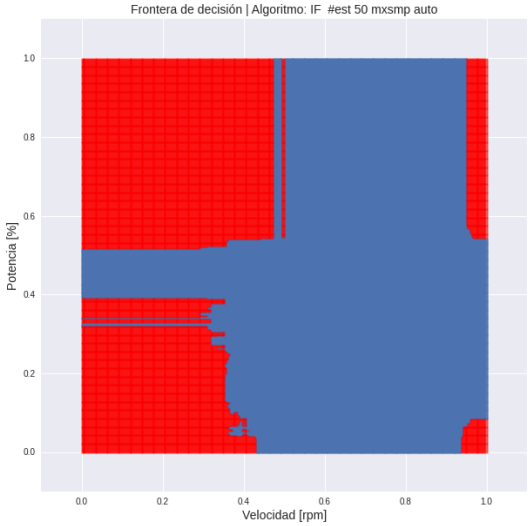
Ventilador 152

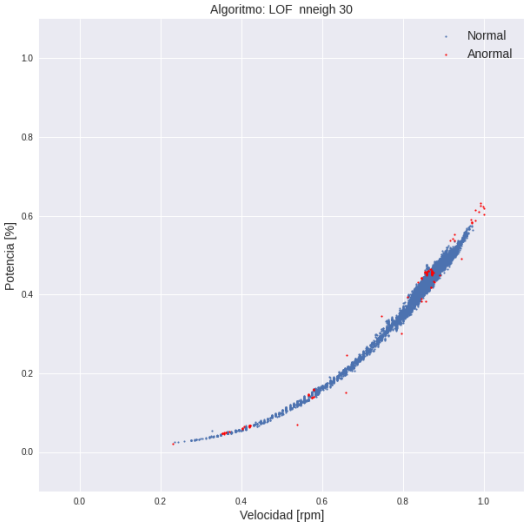


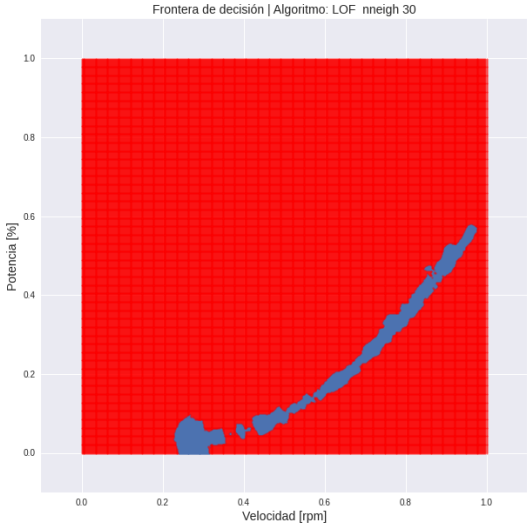


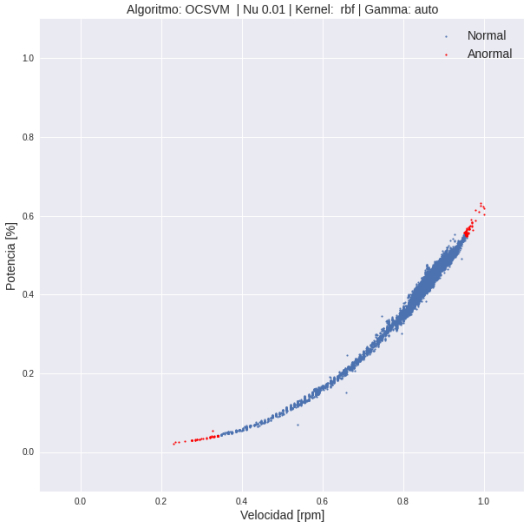


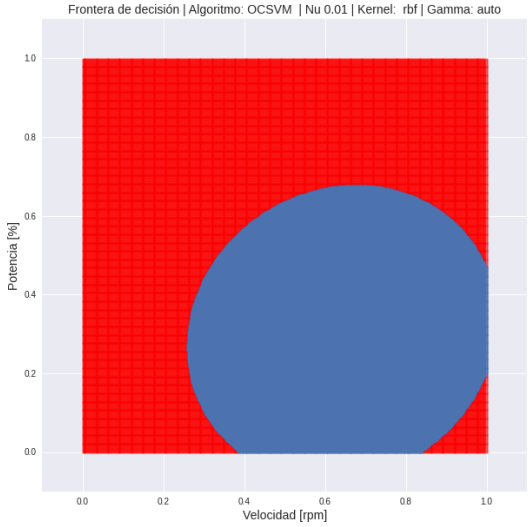

Algoritmo: IF #est 50 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 50 mxsmp auto
Potencia [%]
Velocidad [rpm]

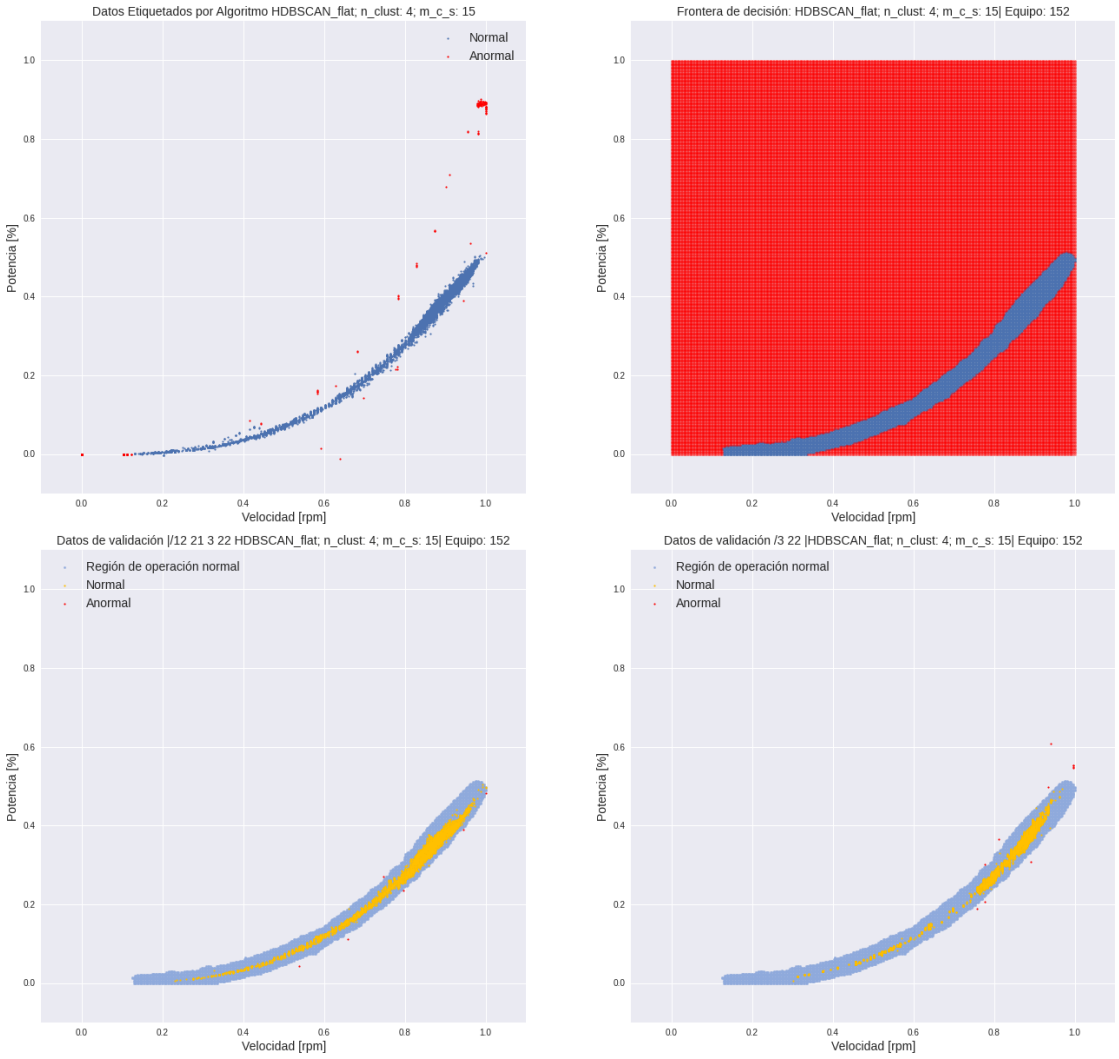

Algoritmo: LOF nneigh 30
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 30
Potencia [%]
Velocidad [rpm]

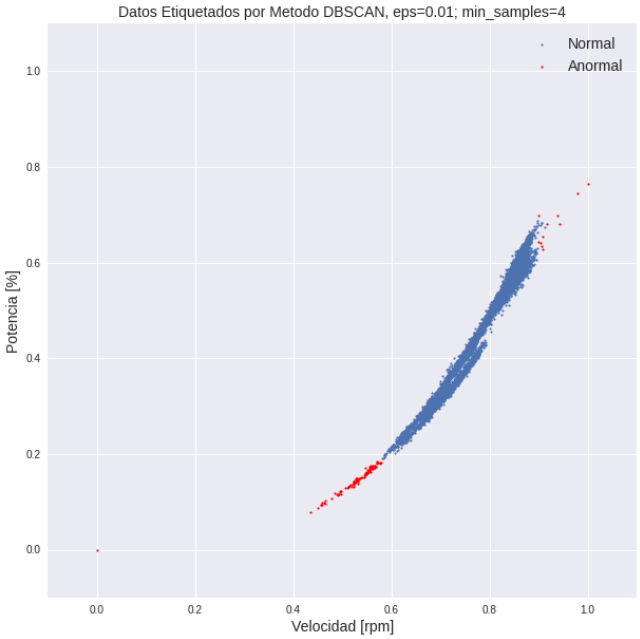

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

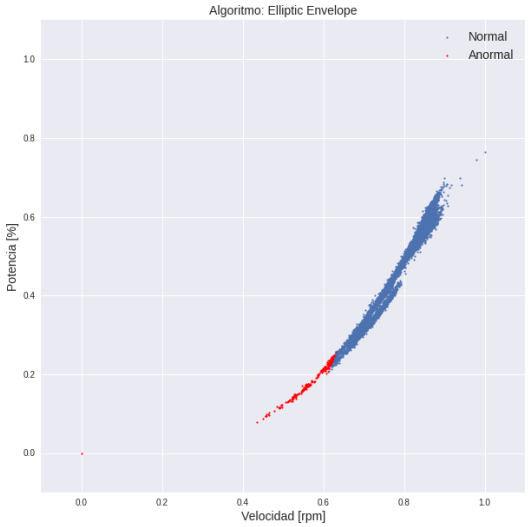

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

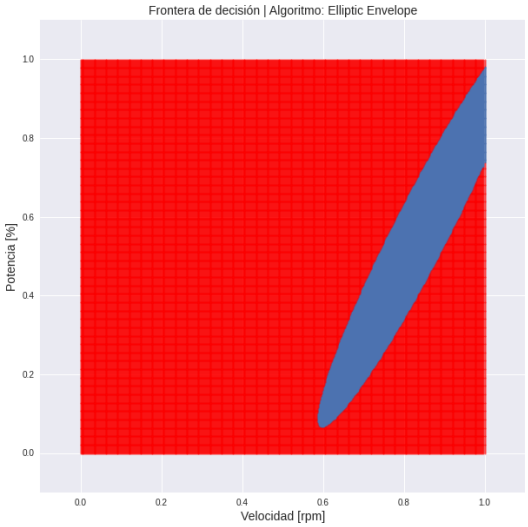
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9891	0.0000	0.0000	0.0000
Isolation Forest	0.9899	0.0609	0.7000	0.1120
Local Outlier Factor	0.9913	0.0952	1.0000	0.1739
OC-SVM	0.9893	0.0579	0.7000	0.1069

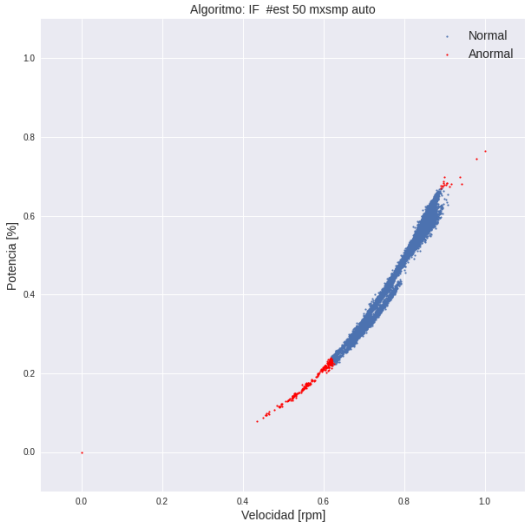


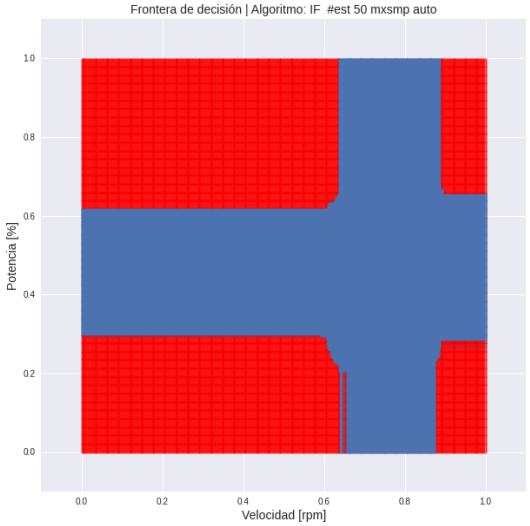
Ventilador 256

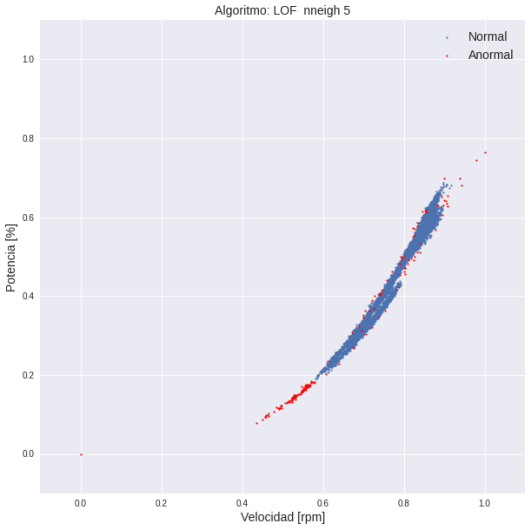


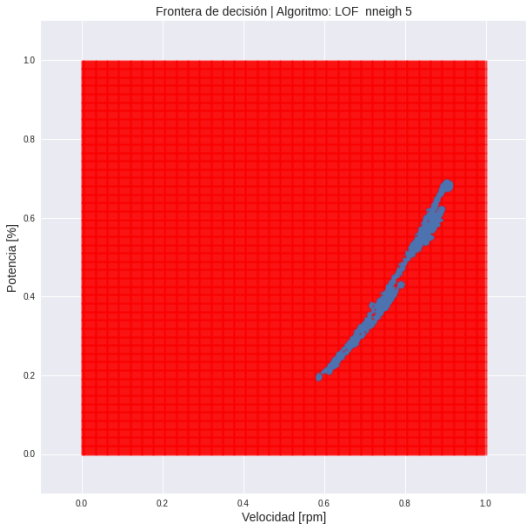


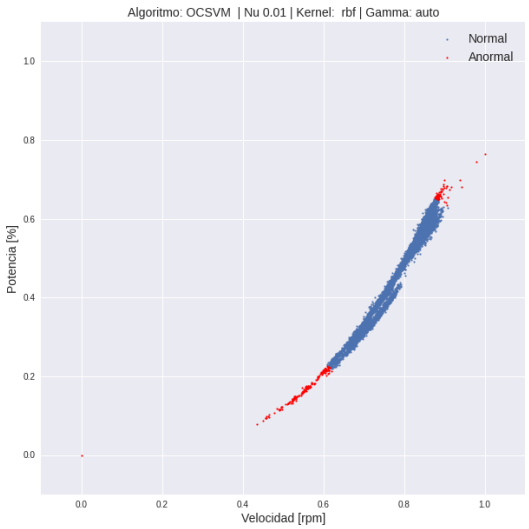


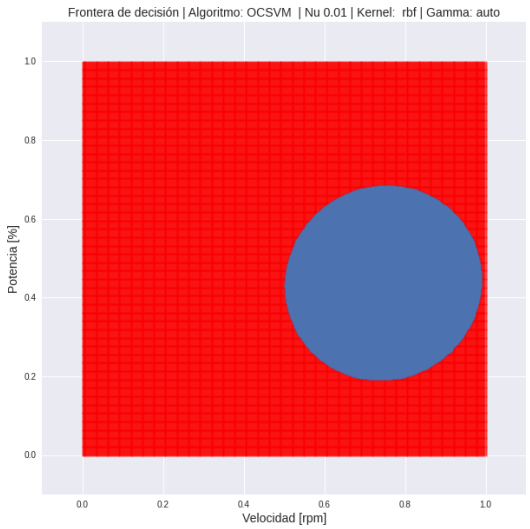

Algoritmo: IF #est 50 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 50 mxsmp auto
Potencia [%]
Velocidad [rpm]

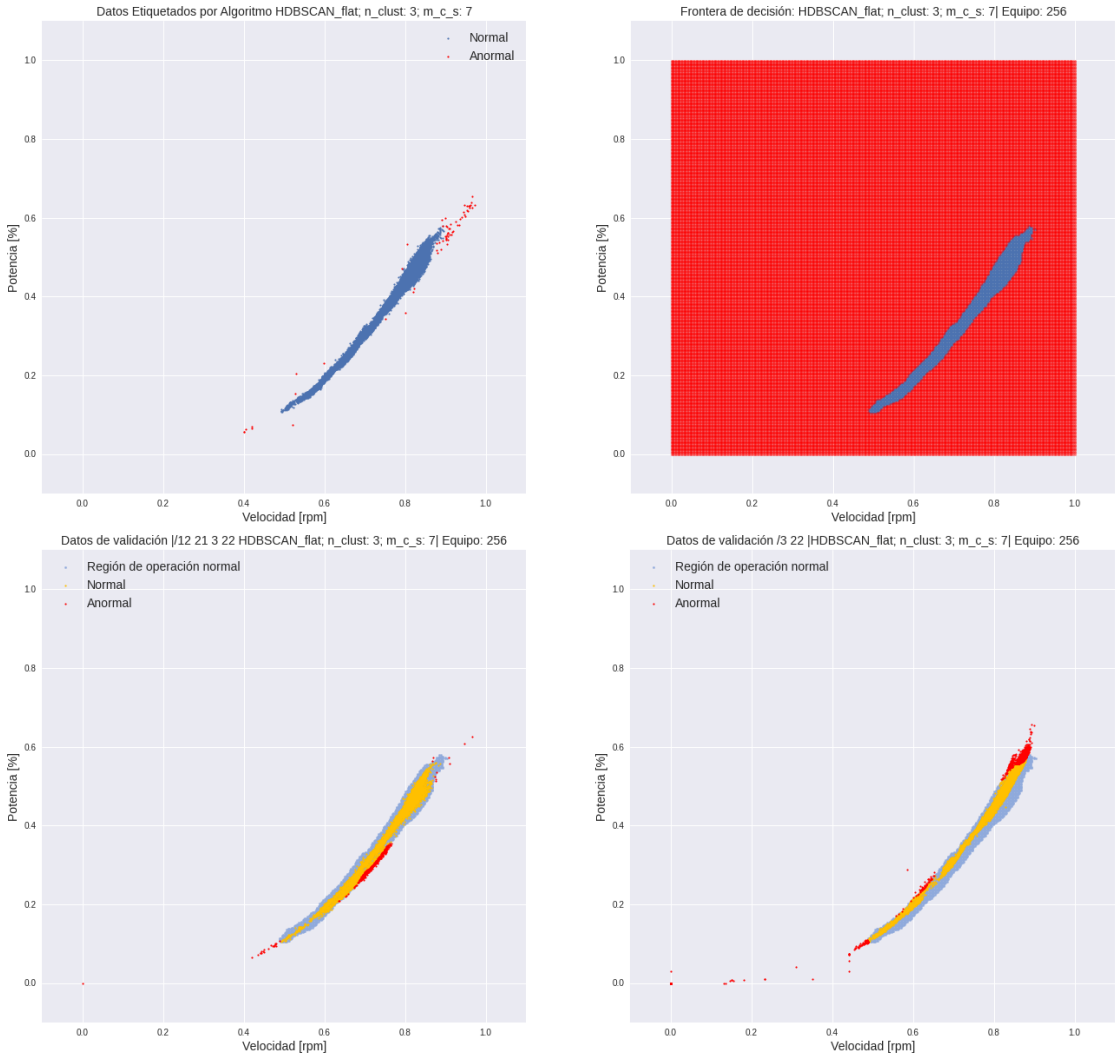

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

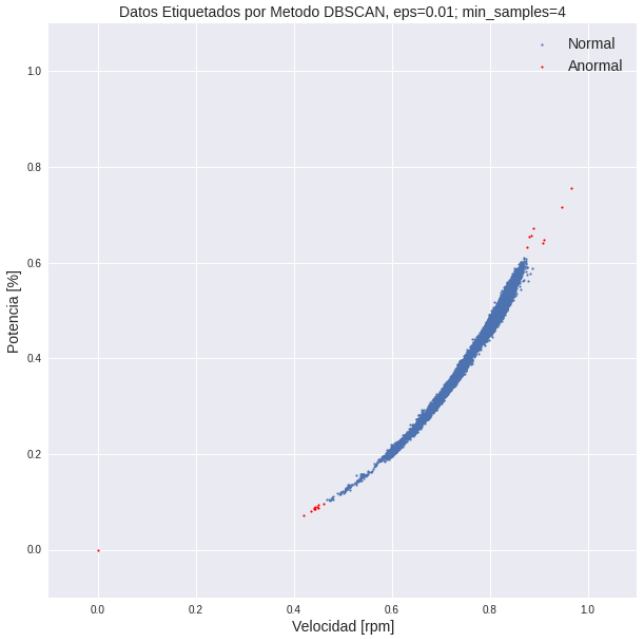

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

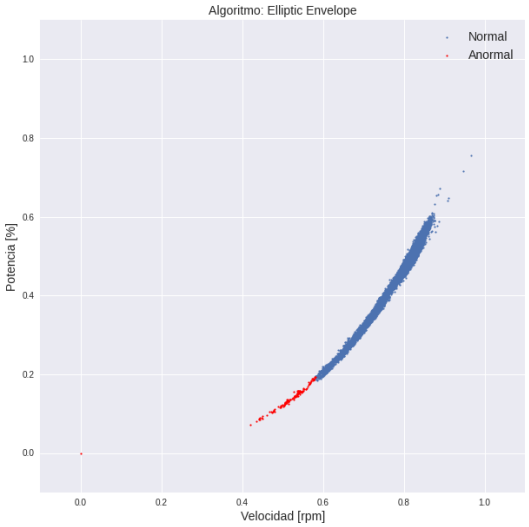

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

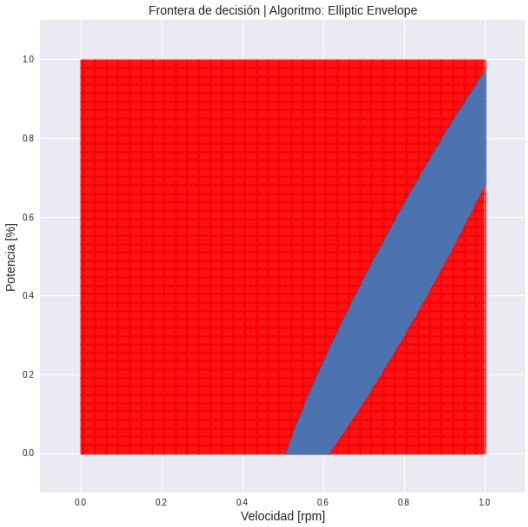
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9892	0.4766	0.9106	0.6257
Isolation Forest	0.9897	0.4917	0.9593	0.6501
Local Outlier Factor	0.9949	0.6667	0.9919	0.7974
OC-SVM	0.9902	0.5021	0.9919	0.6667

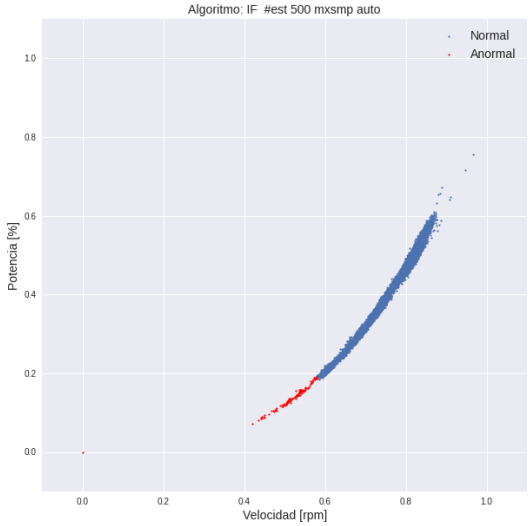


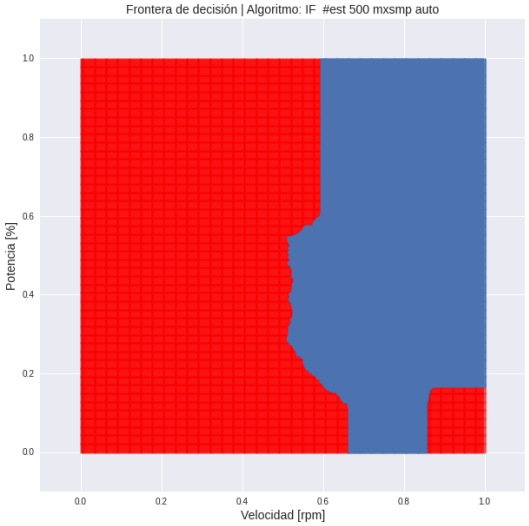
Ventilador 257

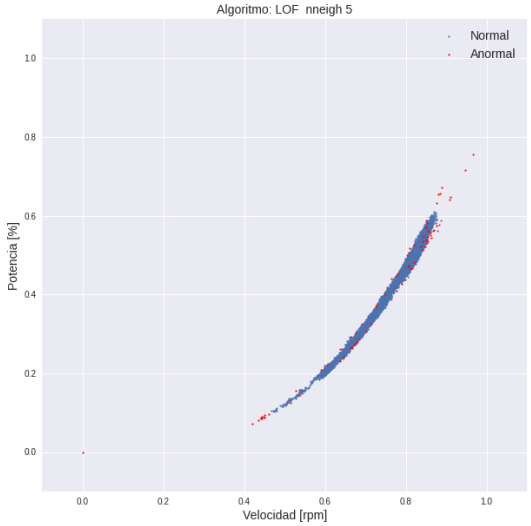


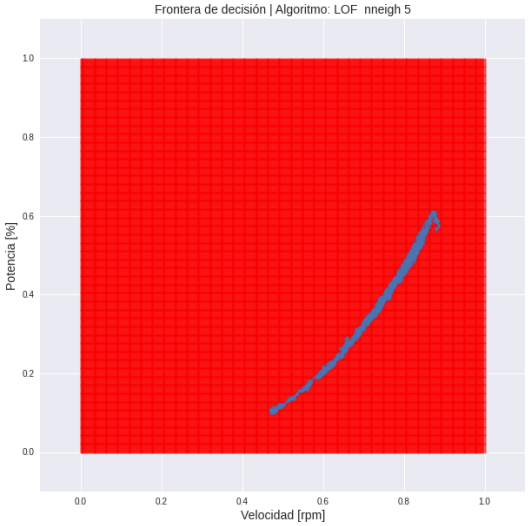


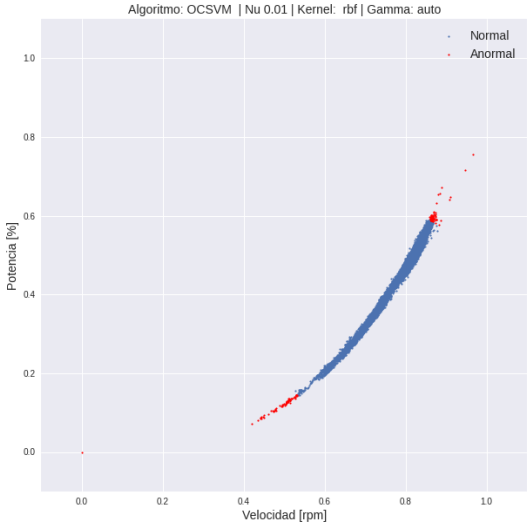


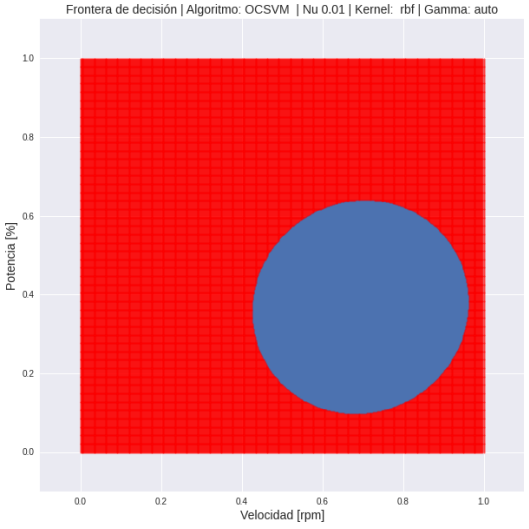

Algoritmo: IF #est 500 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 500 mxsmp auto
Potencia [%]
Velocidad [rpm]

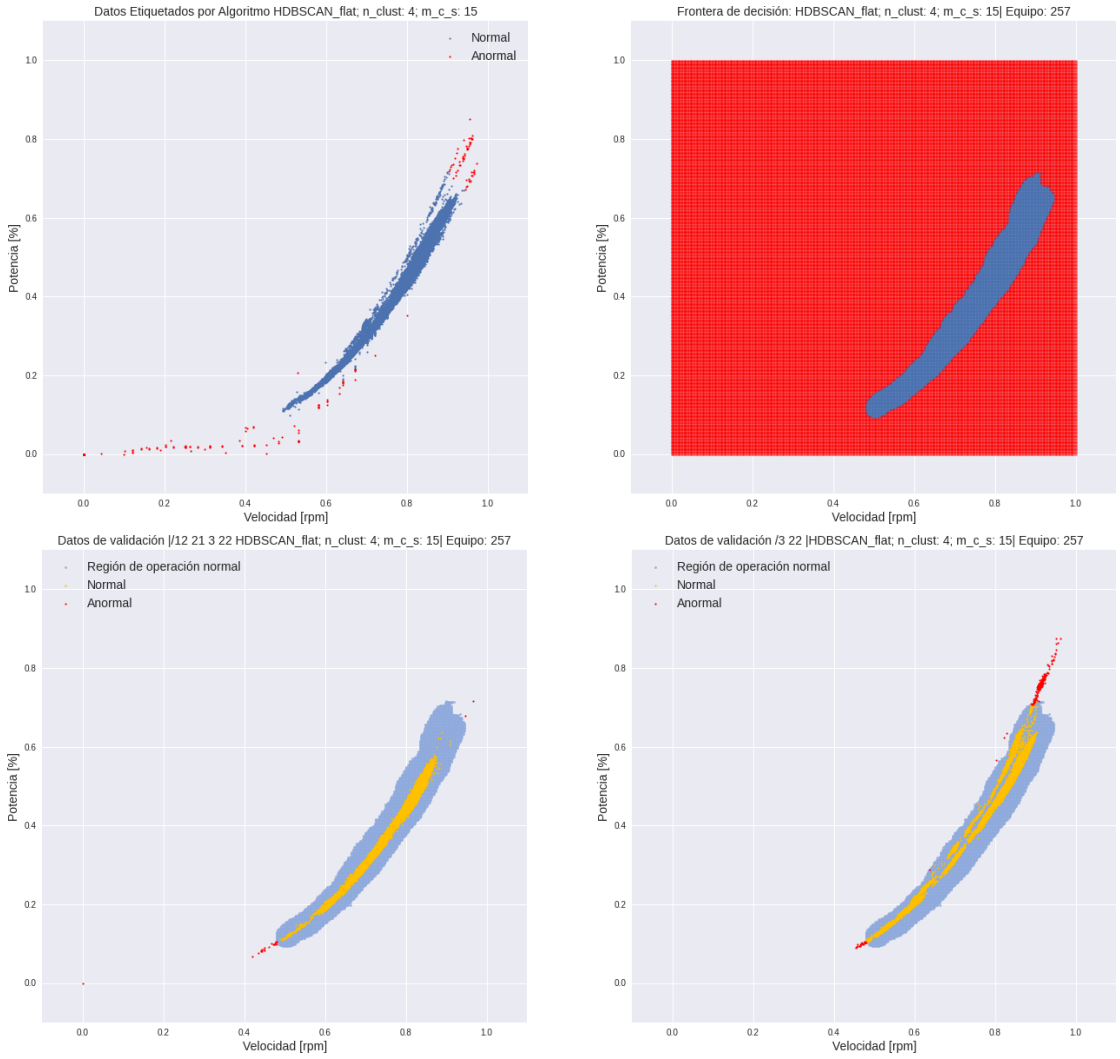

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

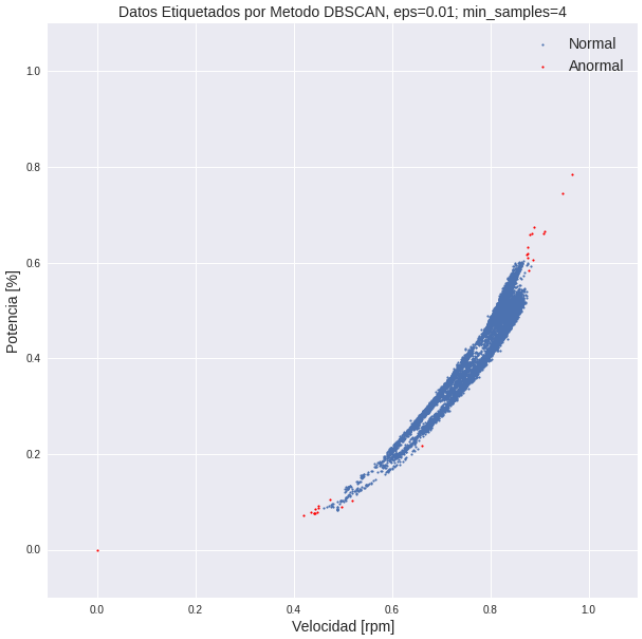

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

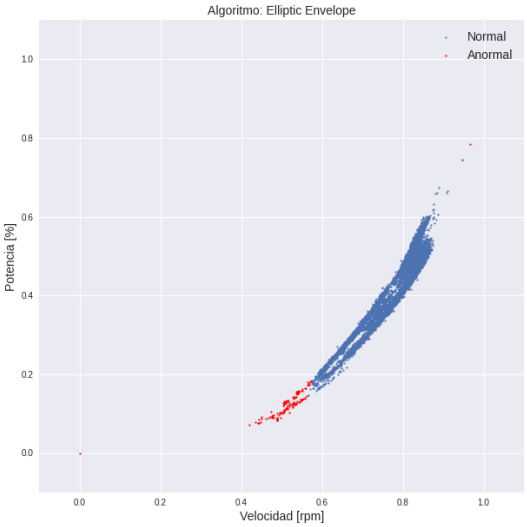

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

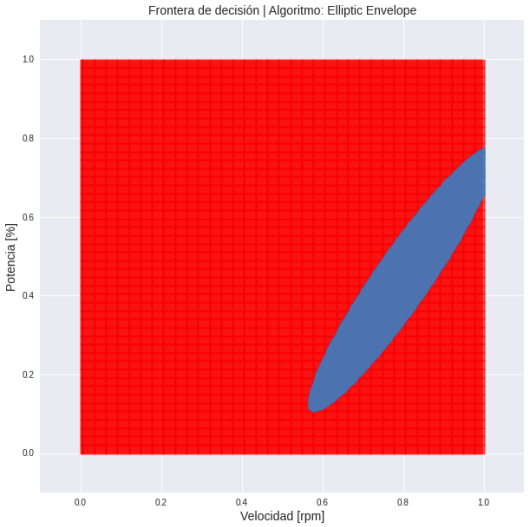
Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9893	0.0815	0.5789	0.1429
Isolation Forest	0.9893	0.0815	0.5789	0.1429
Local Outlier Factor	0.9949	0.2346	1.0000	0.3800
OC-SVM	0.9901	0.1338	1.0000	0.2360

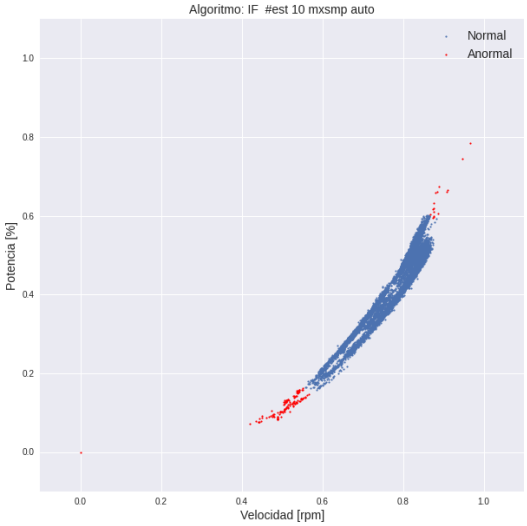


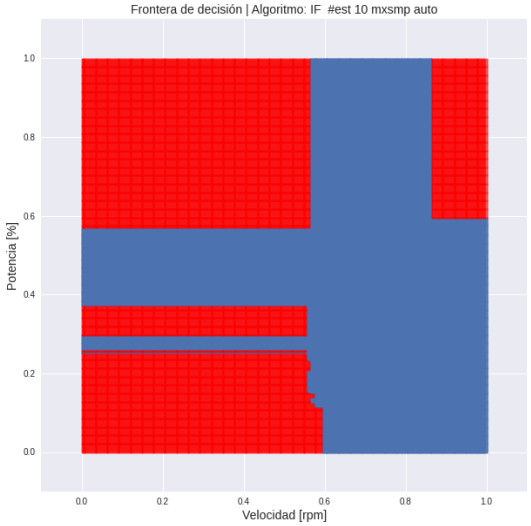
Ventilador 258

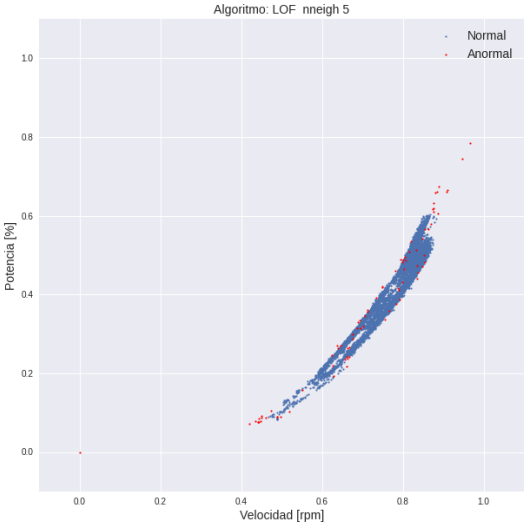


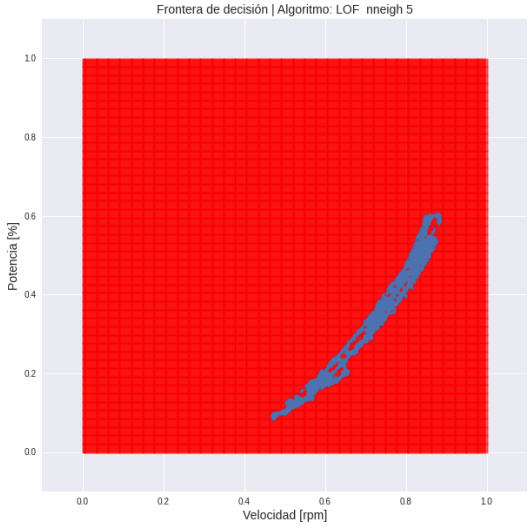


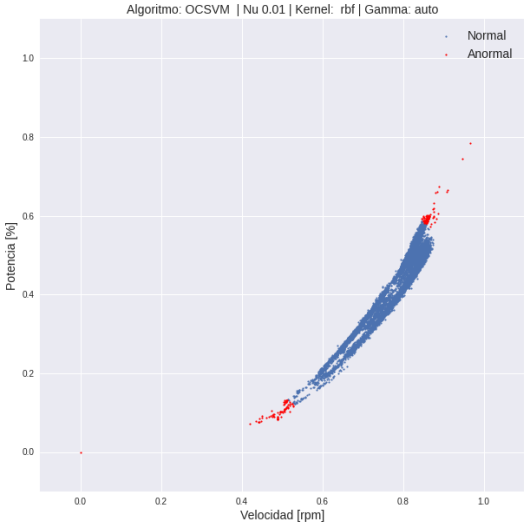


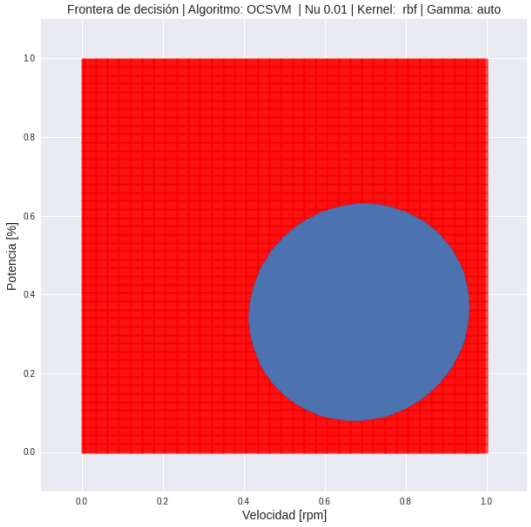

Algoritmo: IF #est 10 mxsmp auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]


Frontera de decisión | Algoritmo: IF #est 10 mxsmp auto
Potencia [%]
Velocidad [rpm]

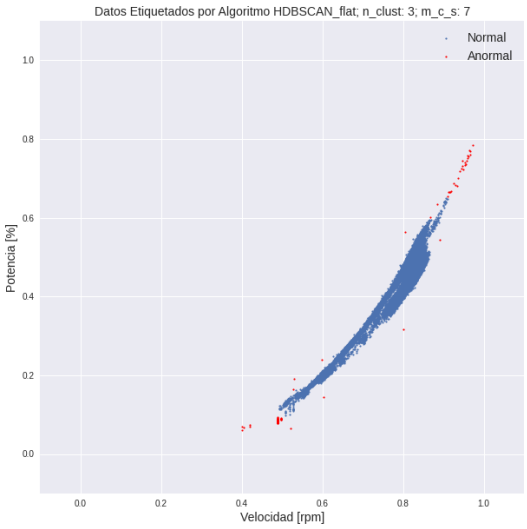

Algoritmo: LOF nneigh 5
Normal
Anormal
Potencia [%]
Velocidad [rpm]

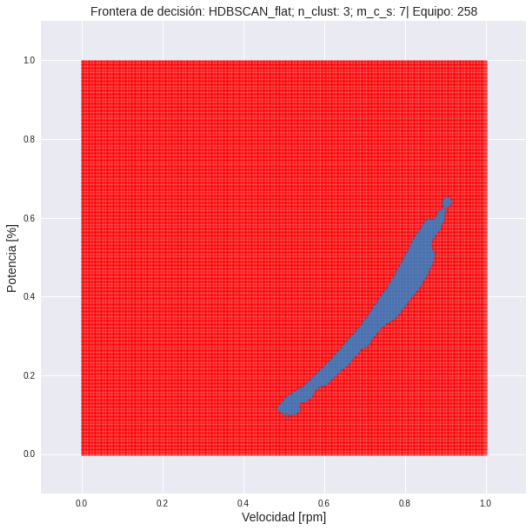

Frontera de decisión | Algoritmo: LOF nneigh 5
Potencia [%]
Velocidad [rpm]

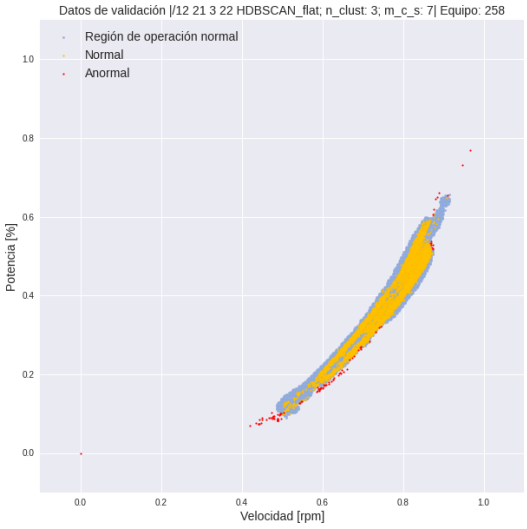

Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Normal
Anormal
Potencia [%]
Velocidad [rpm]

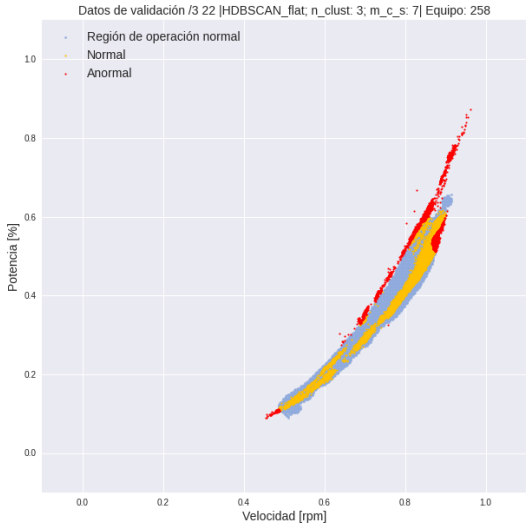

Frontera de decisión | Algoritmo: OCSVM | Nu 0.01 | Kernel: rbf | Gamma: auto
Potencia [%]
Velocidad [rpm]

Algoritmo	Exactitud	Precisión	Sensibilidad	F1-score
Elliptic Envelope	0.9890	0.1079	0.5556	0.1807
Isolation Forest	0.9902	0.1736	0.9259	0.2924
Local Outlier Factor	0.9954	0.3171	0.9629	0.4771
OC-SVM	0.9899	0.1745	0.9629	0.2955









Anexo B

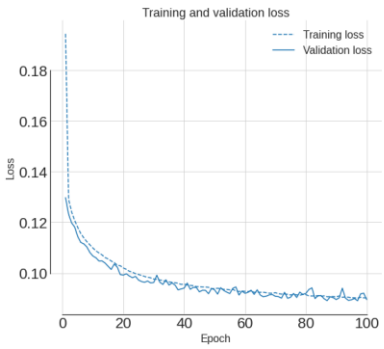
Modelos MLP Iteración II

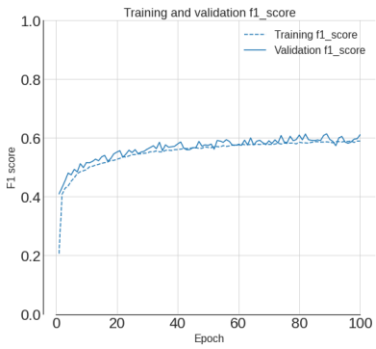
n°	Activación	Dropout	Método balance de clases	Capas ocultas	Etiquetado [min]
1	relu	0.0	NO	(16)(8)	20
2	tanh	0.0	NO	(16)(8)	20
3	relu	0.0	NO	(8)(4)	20
4	relu	0.0	NO	(32)(16)(8)	20
5	relu	0.0	NO	(64)(32)(16)	20
6	relu	0.0	NO	(64)(32)(16)(8)	20
7	tanh	0.0	NO	(32)(16)(8)	20
8	relu	0.5	NO	(32)(16)(8)	20
9	tanh	0.5	NO	(32)(16)(8)	20
10	relu	0.0	Balance pesos	(16)(8)	20
11	relu	0.0	SMOTE	(16)(8)	20
12	selu	0.1 (AlphaDropout)	NO	(16)(8)	20
13	tanh	0.0	NO	(32)(16)(8)	10
14	relu	0.0	NO	(32)(16)(8)	10
15	relu	0.0	NO	(16)(8)	10
16	tanh	0.5	NO	(32)(16)(8)	10
17	relu	0.5	NO	(16)(8)	10
18	relu	0.3	NO	(64)(32)(16)	10

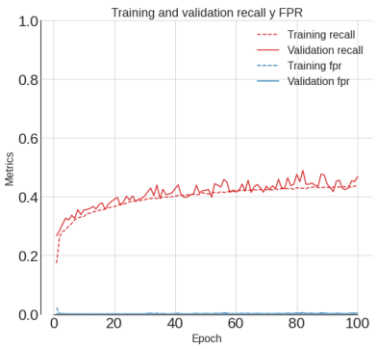
19	relu	0.0	Balance pesos	(32)(16)(8)	10
20	relu	0.5	Balance pesos	(16)(8)	10
21	relu	0.0	SMOTE	(16)(8)(4)	10
22	relu	0.5	SMOTE	(16)(8)	10
23	selu	0.1 (AlphaDropout)	NO	(16)(8)	10
24	relu	0.0	NO	(16)(8)	60
25	relu	0.0	NO	(16)(8)	120
26	relu	0.0	NO	(64)(64)(64)	180
27	relu	0.0	NO	(32)(32)(32)	180
28	relu	0.0	NO	(16)(16)(16)	180
29	relu	0.5	NO	(64)(64)(64)	180
30	relu	0.5	NO	(32)(16)(8)	180
31	relu	0.0	NO	(16)(8)	30
32	relu	0.0	NO	(32)(16)(8)	10
33	relu	0.2	NO	(16)(8)	10
34	tanh	0.0	NO	(16)(8)	10
35	relu	0.0	Balance pesos	(16)(8)	10
36	relu	0.2	Balance pesos	(16)(8)	10
37	relu	0.0	SMOTE	(16)(8)	10
38	relu	0.2	SMOTE	(16)(8)	10

Curvas de aprendizaje mejores modelos

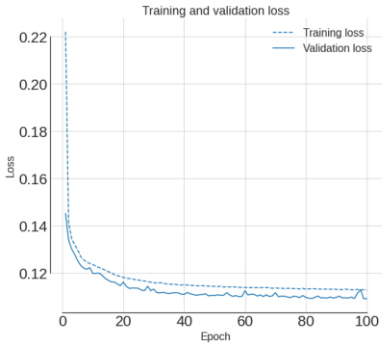
Modelo n°1 | MLP relu | Shift: 20 min

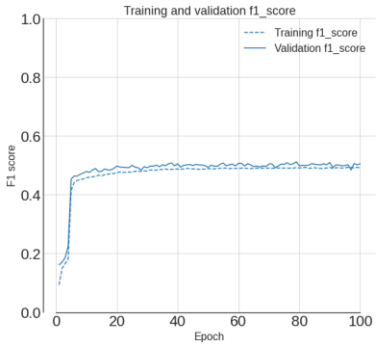


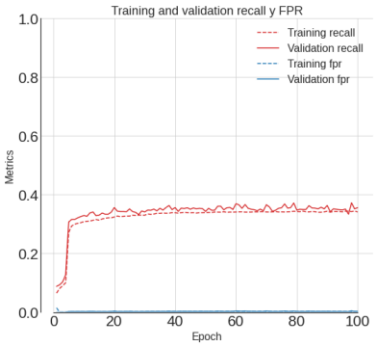




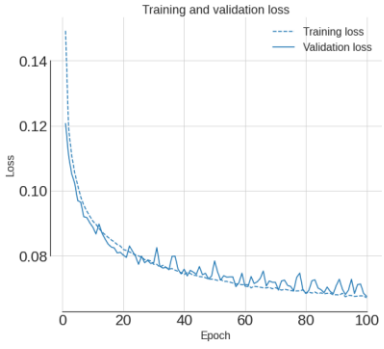
Modelo n°3 | MLP relu | Shift: 20 min | batch: | unidades: 8

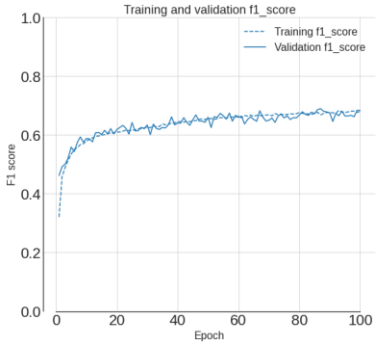




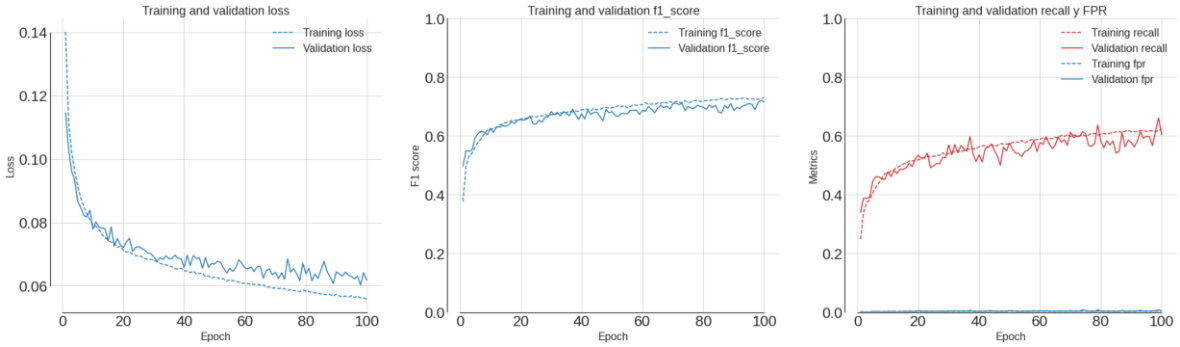


Modelo n°4 | MLP relu | Shift: 20 min | batch: | unidades: 32

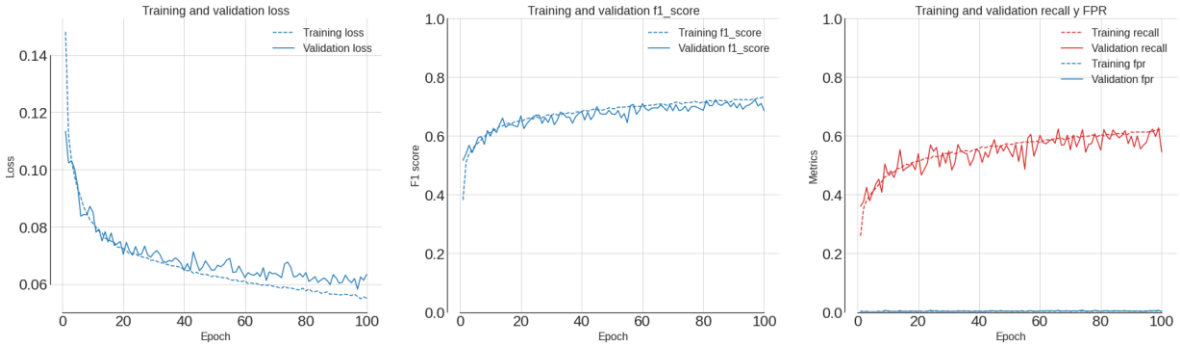




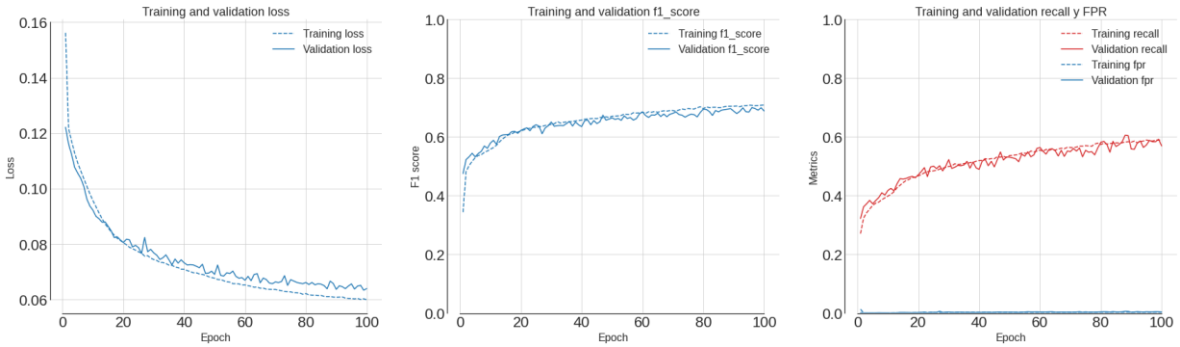
Modelo n°5 | MLP relu | Shift: 20 min | batch: | unidades: 64



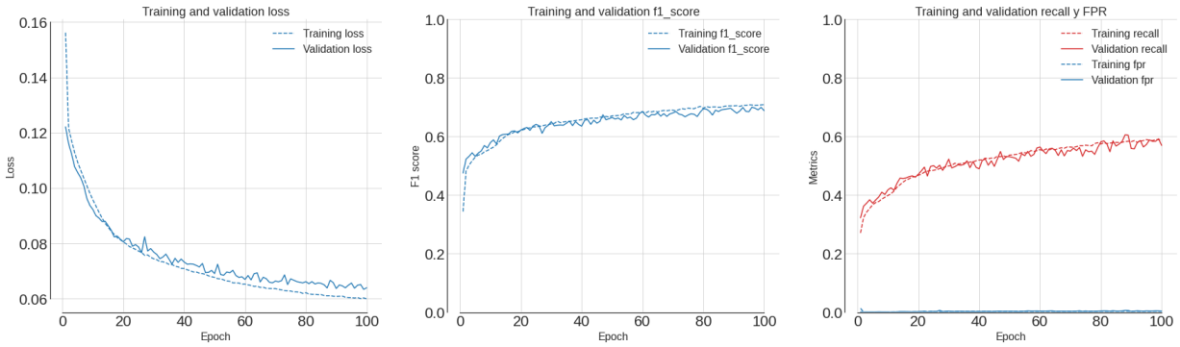
Modelo n°6 | MLP relu | Shift: 20 min | batch: | unidades: 64 | capas: 4



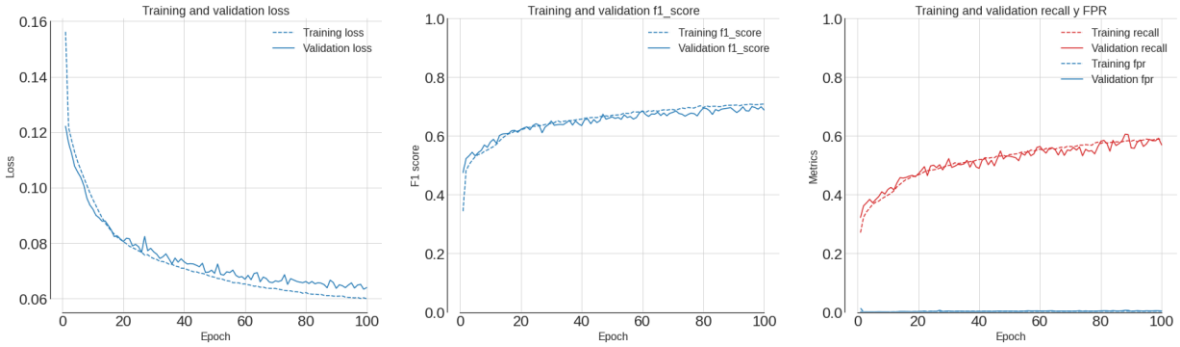
Modelo n°7 | MLP | act: tanh | Shift: 20 min | batch: 128 | unidades: 32 | capas: 3



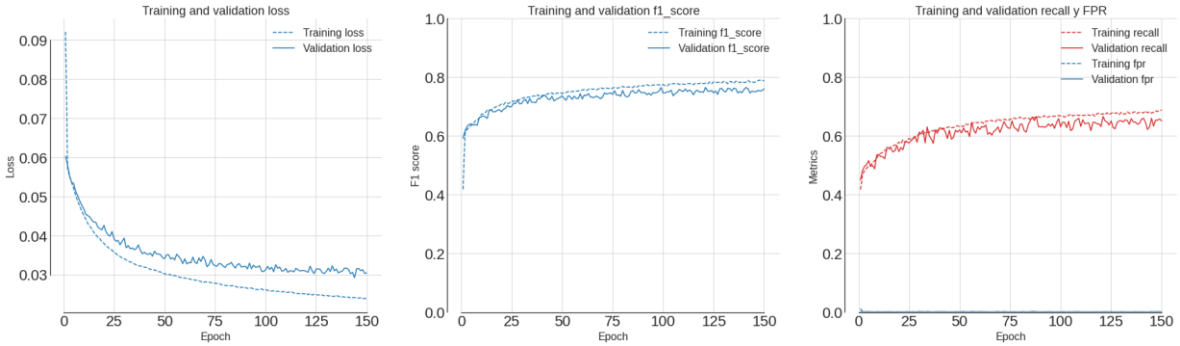
Modelo n°8 | MLP dropout: 0.5 | act: relu | Shift: 20 min | batch: 128 | unidades: 32 | capas: 3



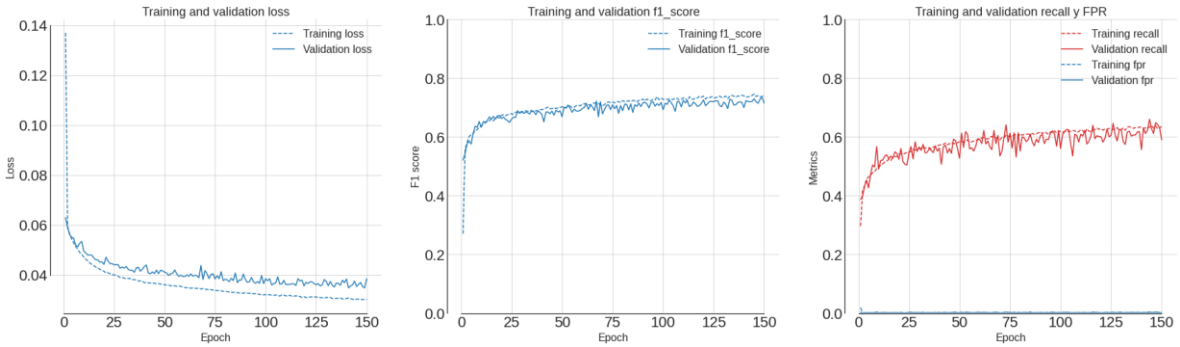
Modelo n°9 | MLP dropout: 0.5 | act: relu | Shift: 20 min | batch: 128 | unidades: 16 | capas: 2



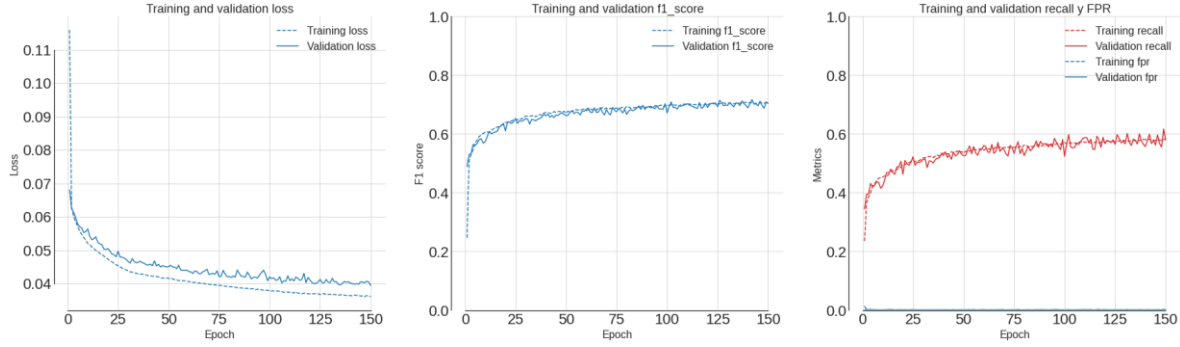
Modelo n°13 | MLP | act: tanh | Shift: 10 min | batch: 128 | unidades: 32 | capas: 3



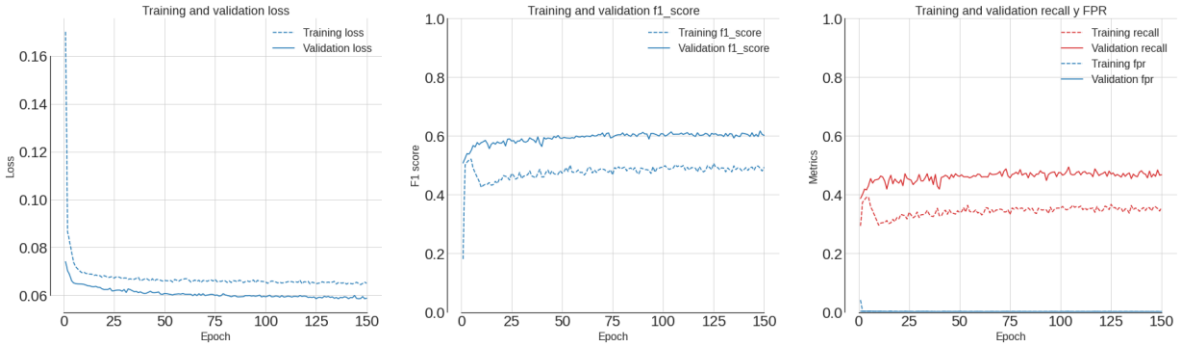
Modelo n°14 | MLP | act: relu | Shift: 10 min | batch: 128 | unidades: 32 | capas: 3



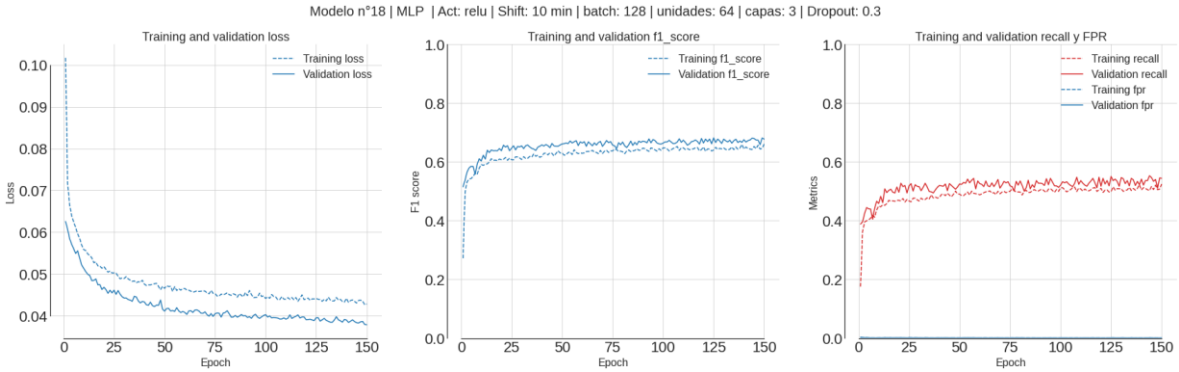
Modelo n°15 | MLP | act: relu | Shift: 10 min | batch: 128 | unidades: 16 | capas: 2



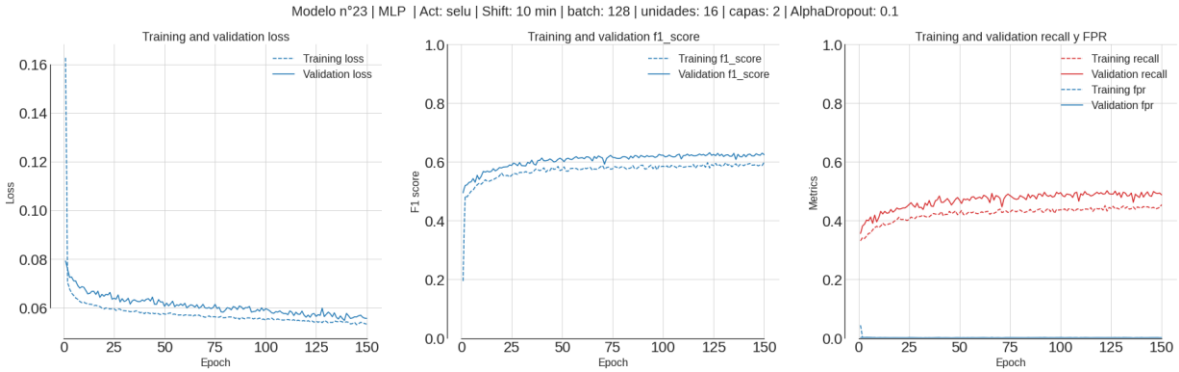
Modelo n°16 | MLP | Act: tanh | Shift: 10 min | batch: 128 | unidades: 32 | capas: 3 | Dropout: 0.5



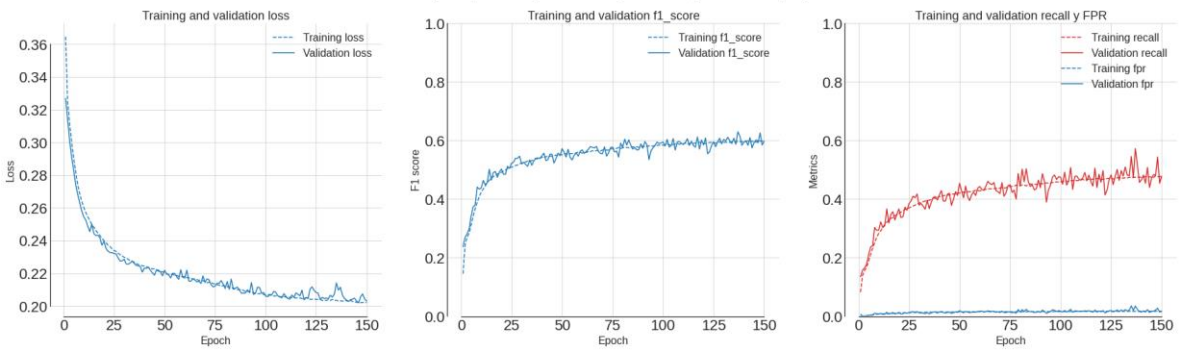
Modelo n°18 | MLP | Act: relu | Shift: 10 min | batch: 128 | unidades: 64 | capas: 3 | Dropout: 0.3



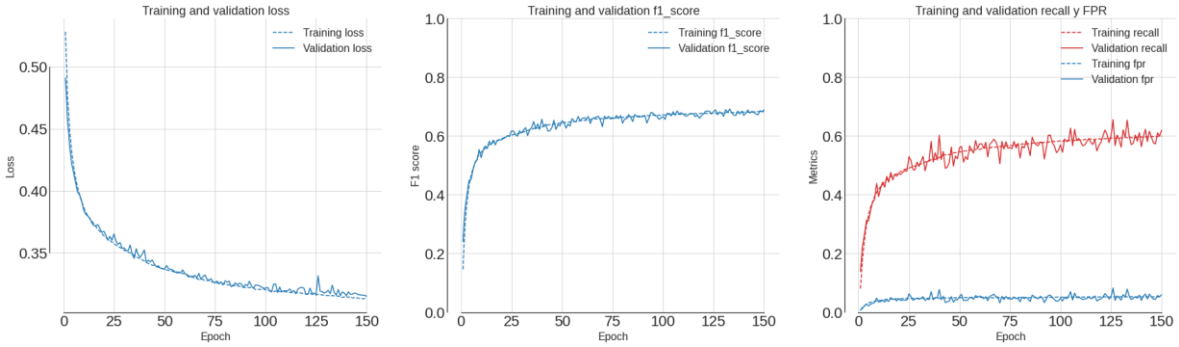
Modelo n°23 | MLP | Act: selu | Shift: 10 min | batch: 128 | unidades: 16 | capas: 2 | AlphaDropout: 0.1



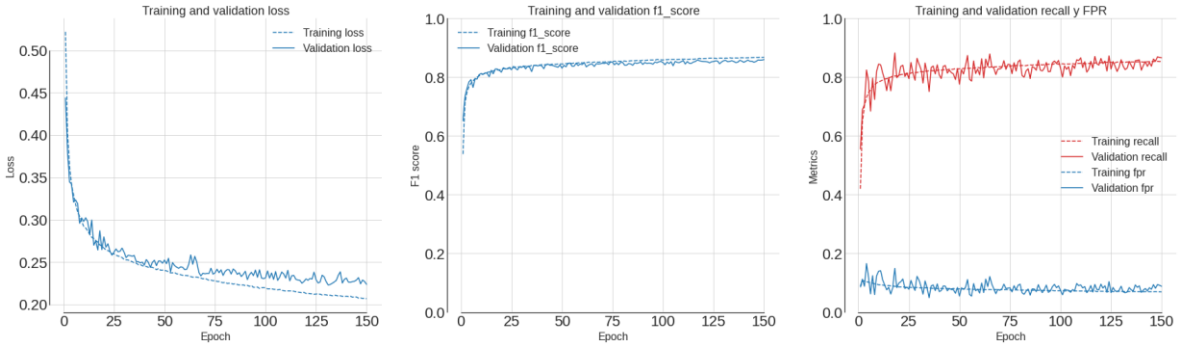
Modelo n°24 | MLP | act: relu | Shift: 60 min | batch: 128 | unidades: 16 | capas: 2



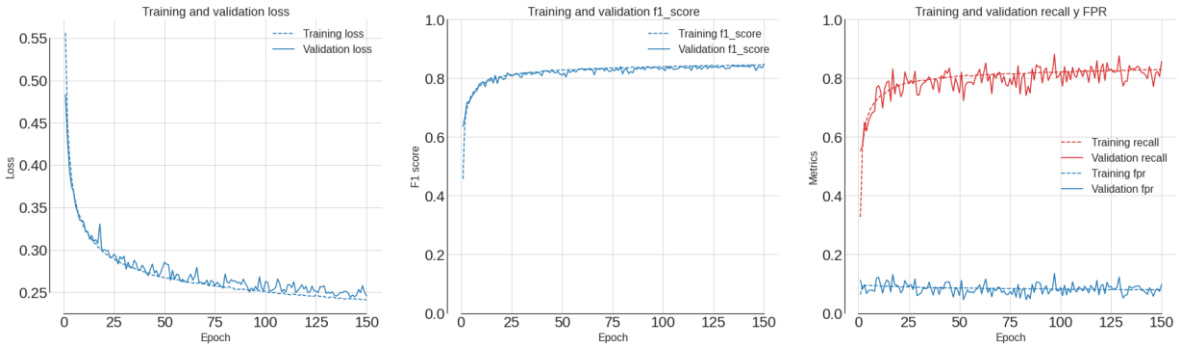
Modelo n°25 | MLP | act: relu | Shift: 120 min | batch: 128 | unidades: 16 | capas: 2



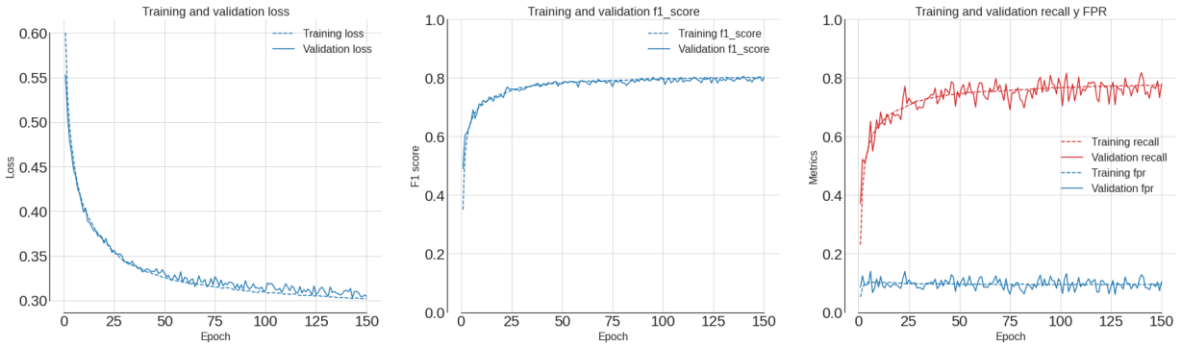
Modelo n°26 | MLP | act: relu | Shift: 180 min | batch: 128 | unidades: 64 | capas: 3 igual tamaño



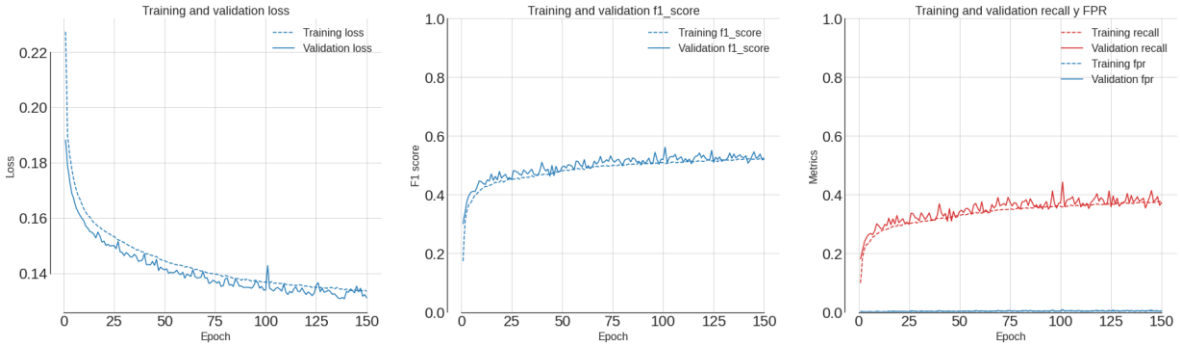
Modelo n°27 | MLP | act: relu | Shift: 180 min | batch: 128 | unidades: 32 | capas: 3 igual tamaño



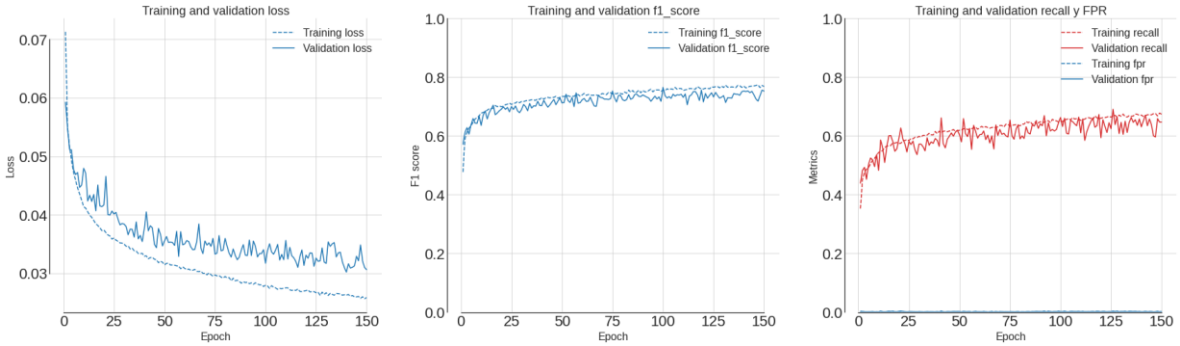
Modelo n°28 | MLP | act: relu | Shift: 180 min | batch: 128 | unidades: 16 | capas: 3 igual tamaño



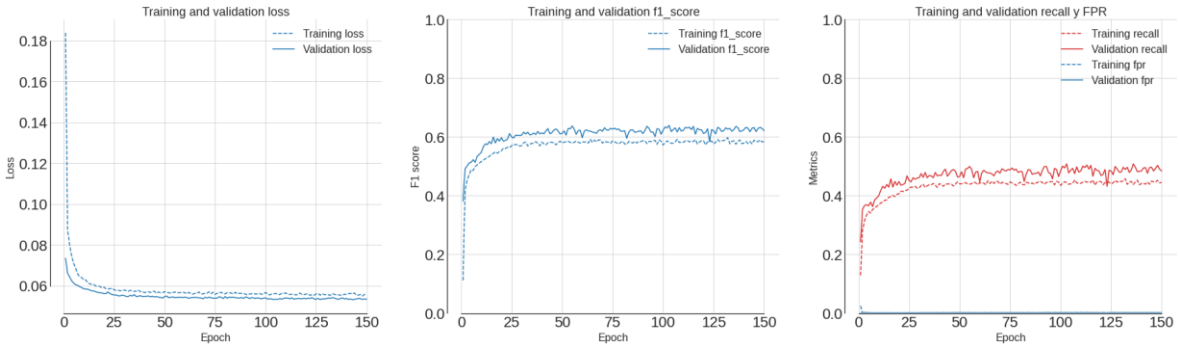
Modelo n°31 | MLP | act: relu | Shift: 30 min | batch: 128 | unidades: 16 | capas: 2 igual tamaño | Dropout: 0



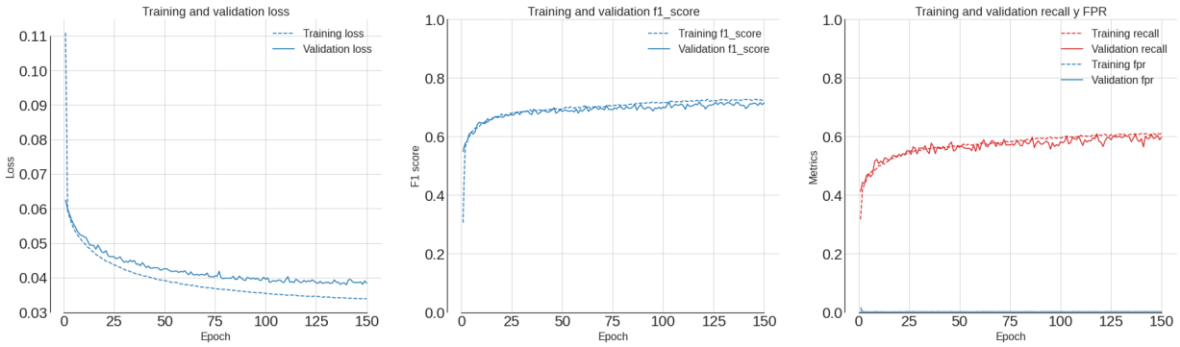
Modelo n°32 | MLP | act: relu | Shift: 10 min | batch: 32 | unidades: 32 | capas: 3 | Dropout: 0 | muestreo 10 seg



Modelo n°33 | MLP | Act: relu | Shift: 10 min | batch: 128 | unidades: 16 | capas: 2 | Dropout: 0.2



Modelo n°34 | MLP | act: tanh | Shift: 10 min | batch: 128 | unidades: 16 | capas: 2 | Dropout: 0 | muestreo 10 seg



Métricas mejores modelos

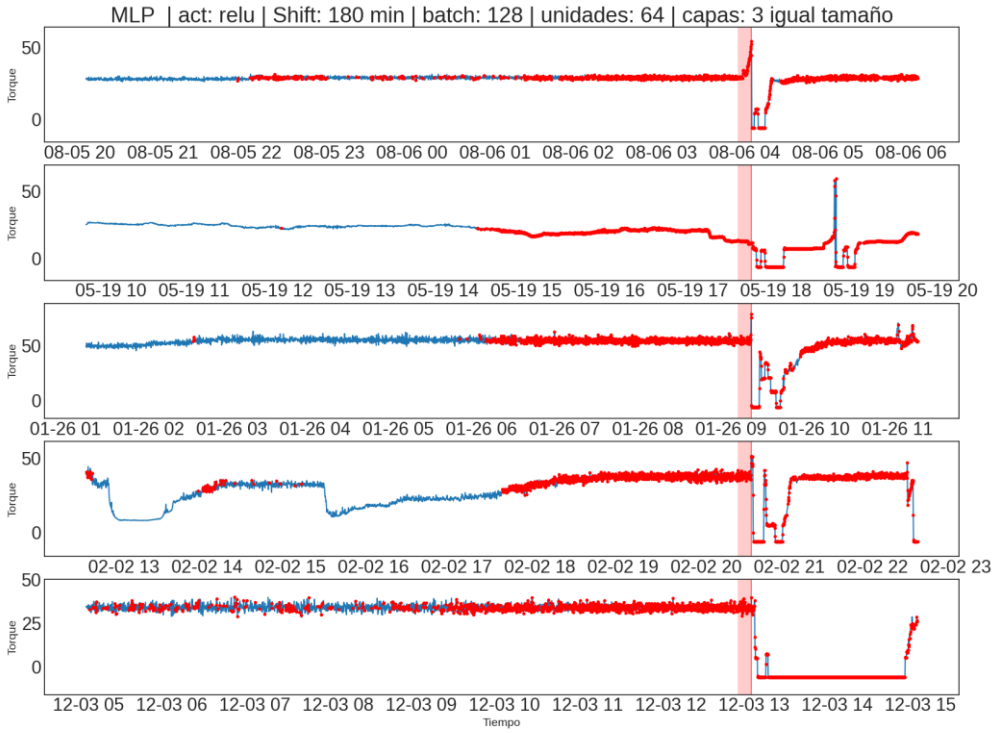
	Loss	Accuracy	Recall	F1_score	FPR
0	0.0780769	0.984598	0.673978	0.652825	0.00858181
1	0.0783389	0.984855	0.542373	0.606128	0.00542931
2	0.0840935	0.983334	0.731805	0.653606	0.0111432
3	0.101597	0.981877	0.765703	0.644836	0.0133762
4	0.0753726	0.984748	0.742772	0.676658	0.00993914
5	0.0663815	0.983269	0.748754	0.657906	0.0115811
6	0.0663815	0.983269	0.748754	0.657906	0.0115811
7	0.0663815	0.983269	0.748754	0.657906	0.0115811
8	0.0386234	0.989439	0.581256	0.702833	0.00159814
9	0.038817	0.989096	0.556331	0.686769	0.00140111
10	0.0377426	0.989546	0.565304	0.699137	0.0011384
11	0.0560656	0.987918	0.490528	0.635659	0.0011603
12	0.255295	0.935477	0.759721	0.335979	0.0606638
13	0.602084	0.811529	0.848455	0.162095	0.189281
14	1.99611	0.640003	0.976072	0.104354	0.367376
15	1.65454	0.637497	0.972084	0.103328	0.36985
16	1.24619	0.660012	0.948156	0.107016	0.346316
17	0.0948372	0.980056	0.65005	0.583445	0.0126976
18	0.0534134	0.987897	0.493519	0.636656	0.00124787
19	0.0372201	0.989675	0.594217	0.712067	0.00164193

Modelo con máxima Sensibilidad en conjunto de testeo.

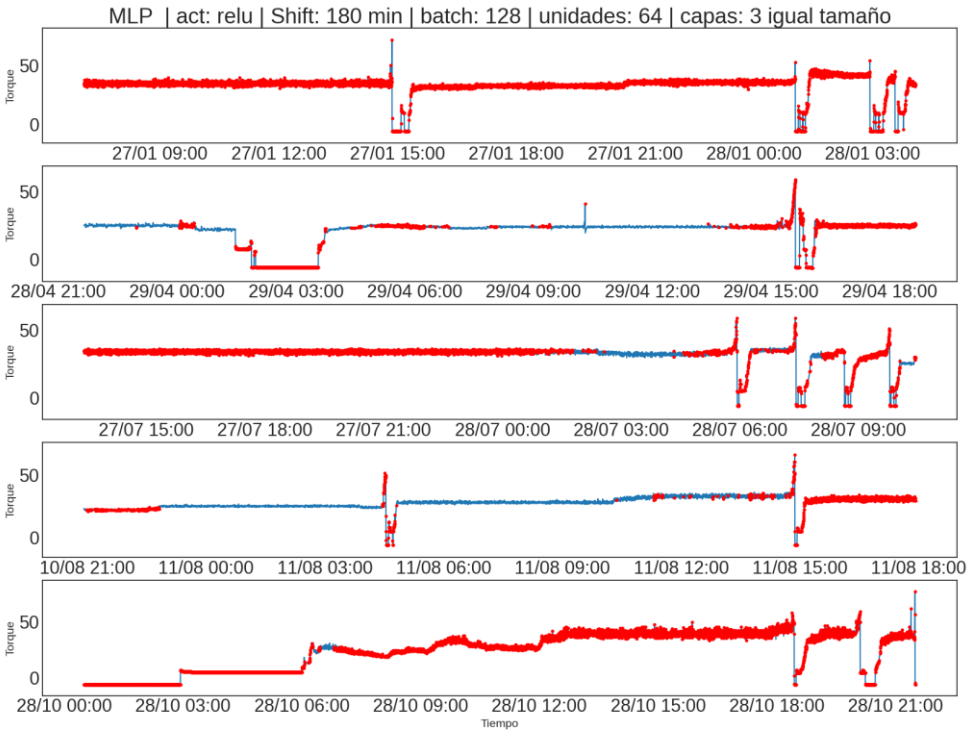
Layer (type)	Output Shape	Param #
hidden_layer_1 (Dense)	(None, 64)	1216
hidden_layer_2 (Dense)	(None, 64)	4160
hidden_layer_3 (Dense)	(None, 64)	4160
output_layer (Dense)	(None, 1)	65

Total params: 9,601
Trainable params: 9,601
Non-trainable params: 0

Eventos conocidos



Eventos no conocidos

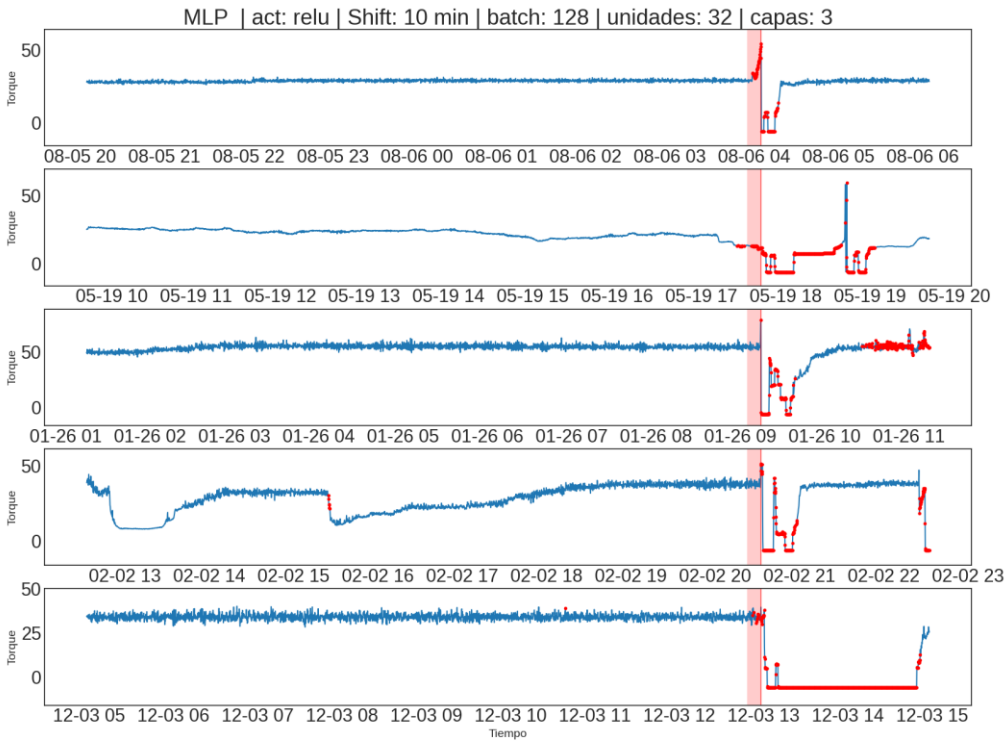


Modelo con pérdida mínima, Exactitud máxima y F1-score máximo en conjunto de testeo.

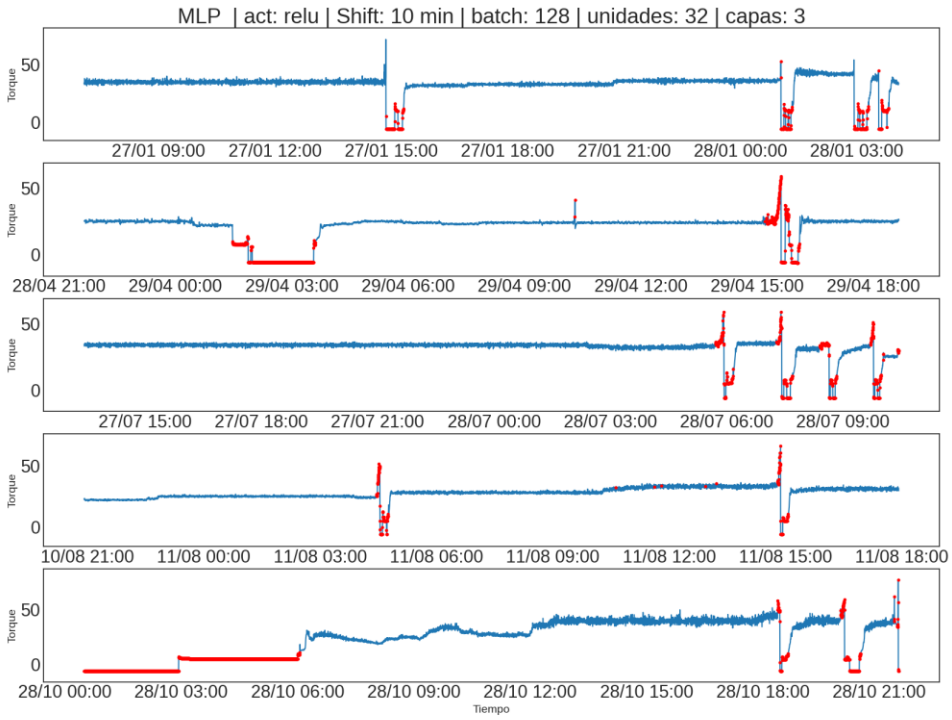
Layer (type)	Output Shape	Param #
hidden_layer_1 (Dense)	(None, 32)	608
hidden_layer_2 (Dense)	(None, 16)	528
hidden_layer_3 (Dense)	(None, 8)	136
output_layer (Dense)	(None, 1)	9

Total params: 1,281
Trainable params: 1,281
Non-trainable params: 0

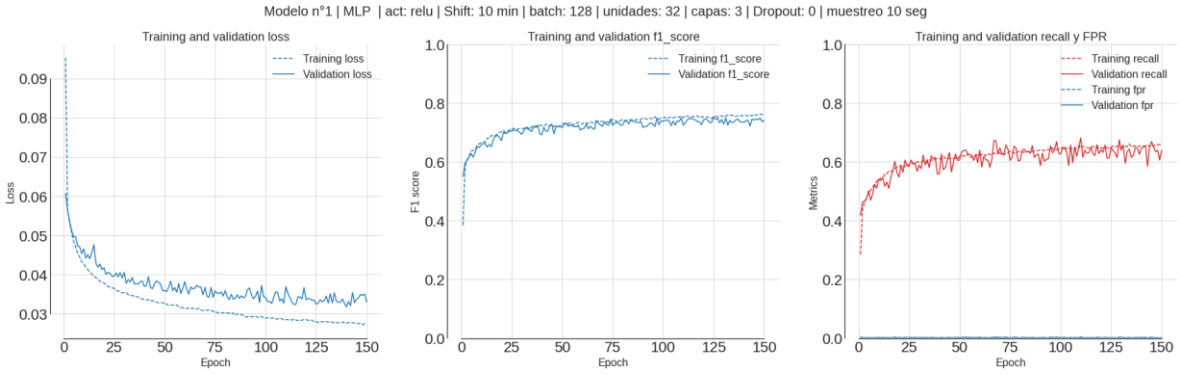
Eventos conocidos



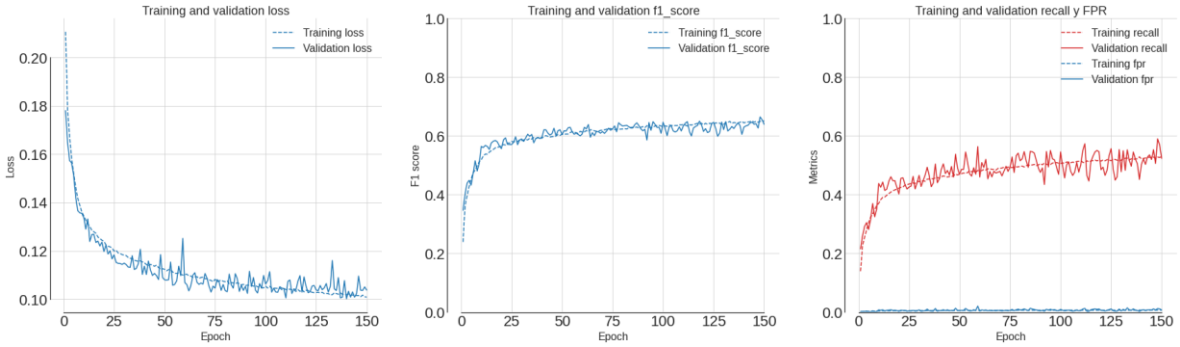
Eventos no conocidos



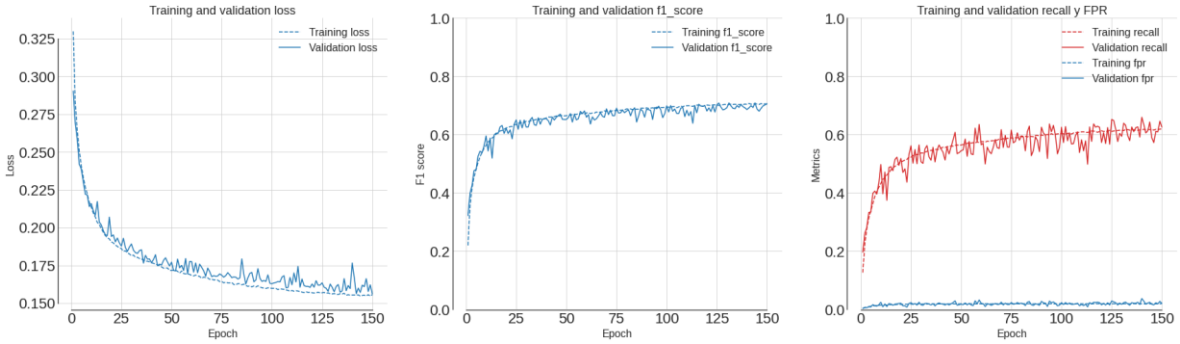
Curvas de aprendizaje modelos MLP iteración III



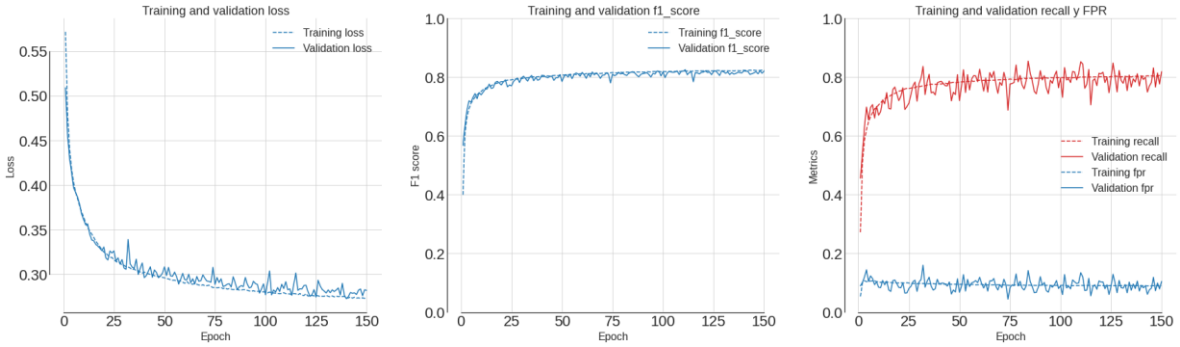
Modelo n°2 | MLP | act: relu | Shift: 30 min | batch: 128 | unidades: 32 | capas: 3 | Dropout: 0 | muestreo 10 seg



Modelo n°3 | MLP | act: relu | Shift: 60 min | batch: 128 | unidades: 32 | capas: 3 | Dropout: 0 | muestreo 10 seg



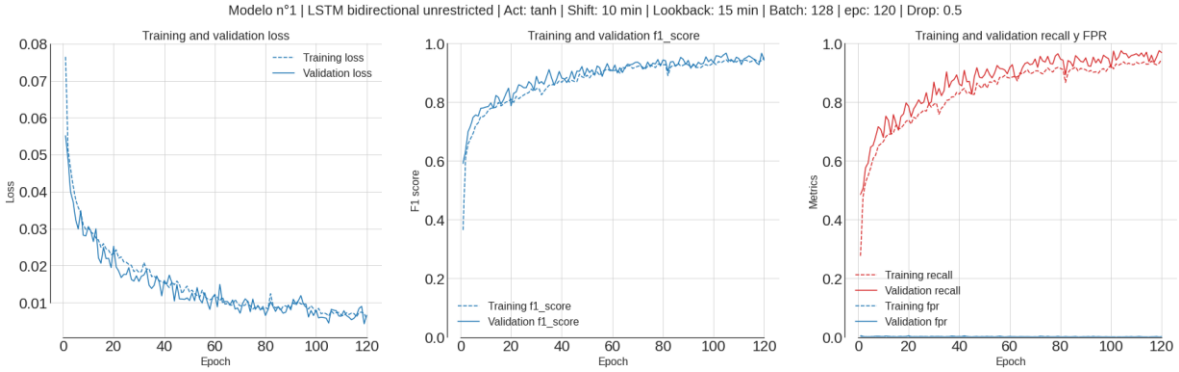
Modelo n°4 | MLP | act: relu | Shift: 180 min | batch: 128 | unidades: 32 | capas: 3 | Dropout: 0 | muestreo 10 seg

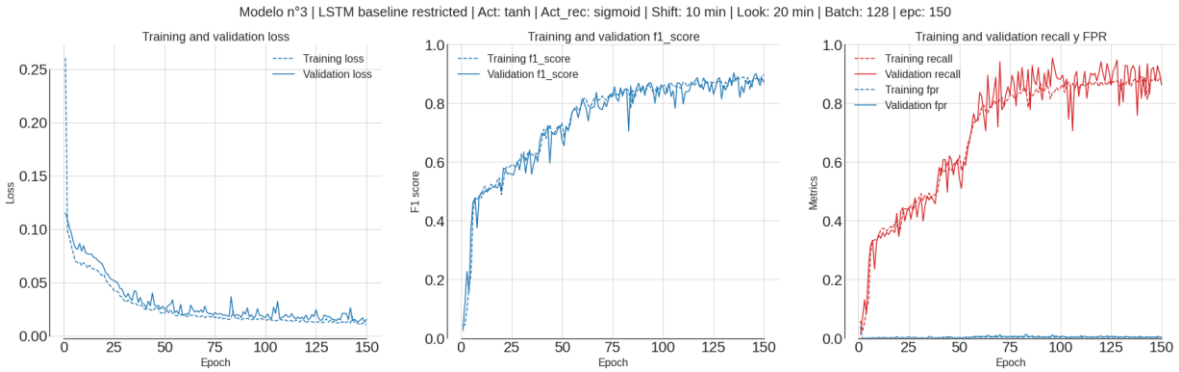


Modelos LSTM Versión I: Iteración II

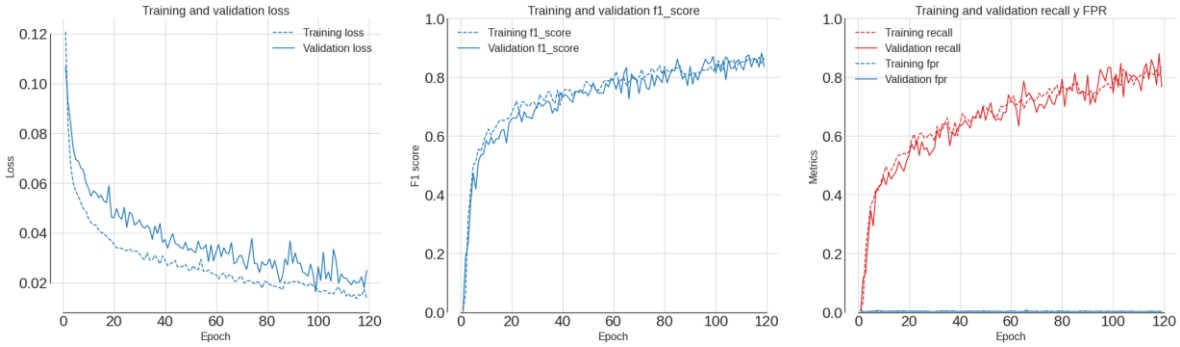
n°	Activación	Etiquetas [min]	Ventana [min]	Dropout / Drop_Rec	Capas LSTM	Estructura
1	tanh	10	15	0.5/0.0	(16)(8)	Unrestricted Stateless Bidirectional
2	relu	10	30	0.5/0.5	(16)(8)	Unrestricted Stateless
3	tanh	10	20	0.0/0.0	(16)(8)	Restricted Stateless
4	tanh	30	60	0.0/0.0	(16)(8)	Restricted Stateless
5	tanh	30	60	0.5/0.0	(16)(8)	Restricted Stateless
6	tanh	60	180	0.0/0.0	(32)(16)(8)	Restricted Stateless
7	relu	60	180	0.0/0.0	(32)(16)(8)	Restricted Stateless
8	tanh	10	60	0.5/0.0	(16)(8)	Unrestricted Stateless Bidirectional
9	tanh	10	60	0.5/0.5	(16)(8)	Unrestricted Stateless Bidirectional
10	tanh	10	30	0.0/0.0	(16)(16)	Restricted Stateless
11	relu	10	30	0.0/0.0	(16)(8)	Restricted Stateless
12	tanh	10	30	0.0/0.0	(16)(8)	Unrestricted Stateless
13	tanh	10	30	0.5/0.5	(16)(8)	Unrestricted Stateless
14	tanh	10	30	0.0/0.0	(16)(8)	Unrestricted Stateless Backwards
15	tanh	10	30	0.5/0.0	(16)(8)	Unrestricted Stateless Bidirectional
16	tanh	10	30	0.0/0.0	(16)(8)	Unrestricted Statefull

Curvas de aprendizaje modelos seleccionados

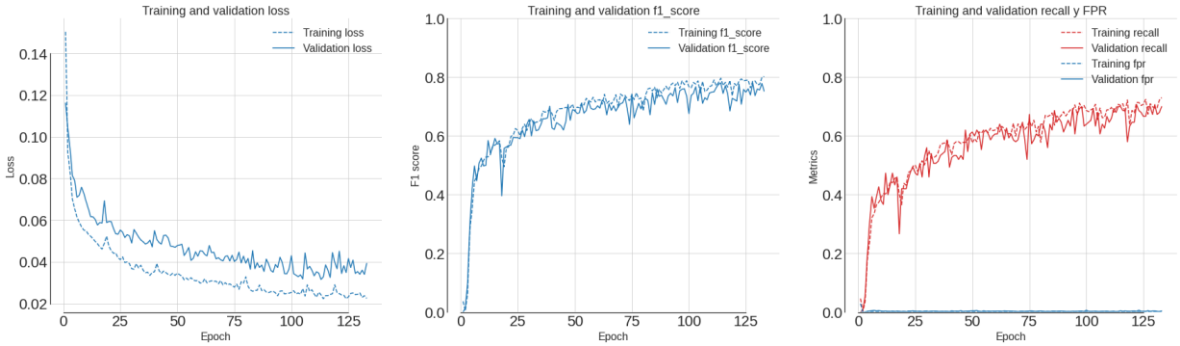




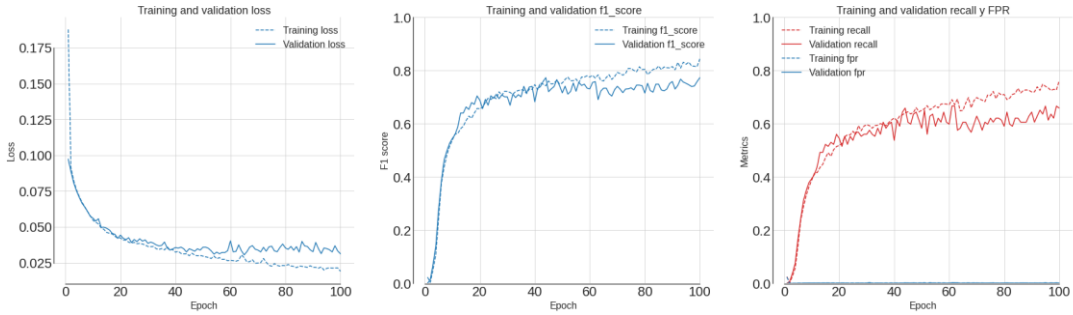
Modelo n°8 | LSTM bidirectional unrestricted | Act: tanh | Act_rec: sigmoid | Shift: 10 min | Lookback: 60 min | Batch: 64 | dropout: 0.5 | muestreo 1 min



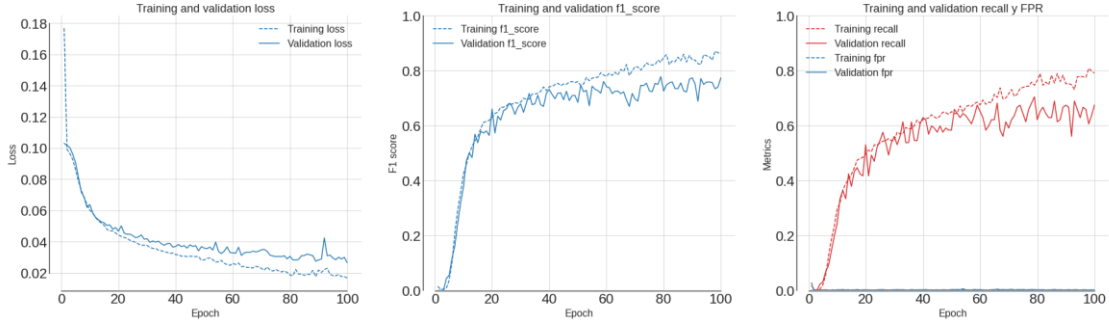
Modelo n°9 | LSTM bidirectional unrestricted | Act: tanh | Act_rec: sigmoid | Shift: 10 min | Lookback: 60 min | Batch: 128 | dropout: 0.5 | muestreo 1 min | capas: 2



Modelo n°10 | LSTM 2 RESTRICTED STATELESS BASELINE | V.I | Act: tanh | Shift: 10 min | Lookback: 30 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

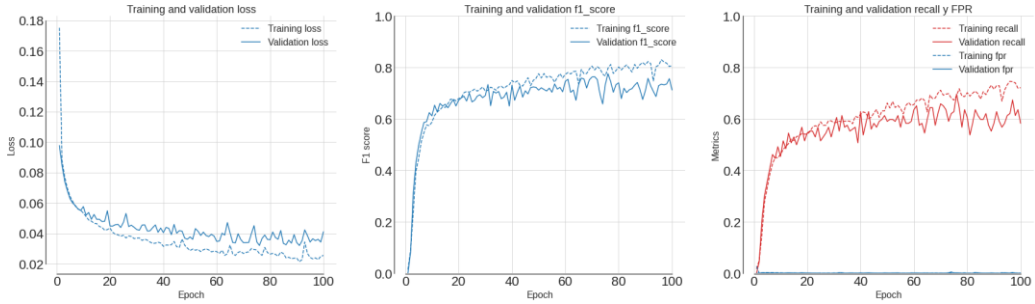


Modelo n°12 | LSTM 2 UNRESTRICTED STATELESS | V.I | Act: tanh | Shift: 10 min | Lookback: 30 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

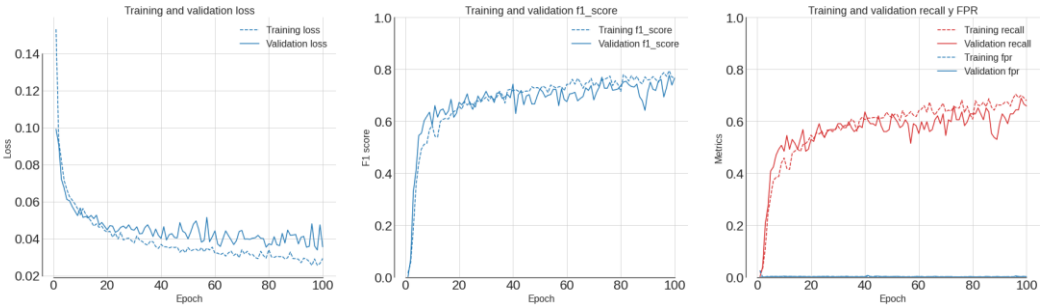


Modelo n°13 | LSTM 2 UNRESTRICTED STATELESS | V.I | Act: tanh | Shift: 10 min | Lookback: 30 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0.5 | drop: 0.5 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

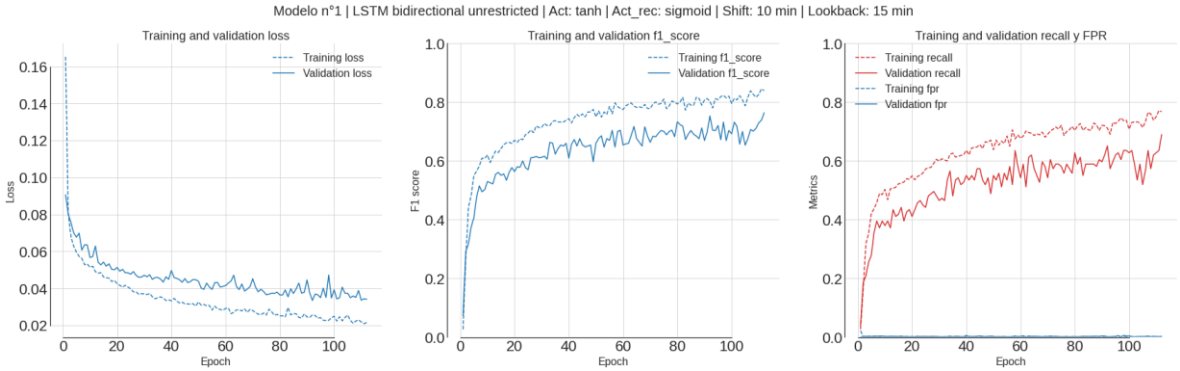
Modelo n°14 | LSTM 2 UNRESTRICTED STATELESS BACKWARDS | V.I | Act: tanh | Shift: 10 min | Lookback: 30 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

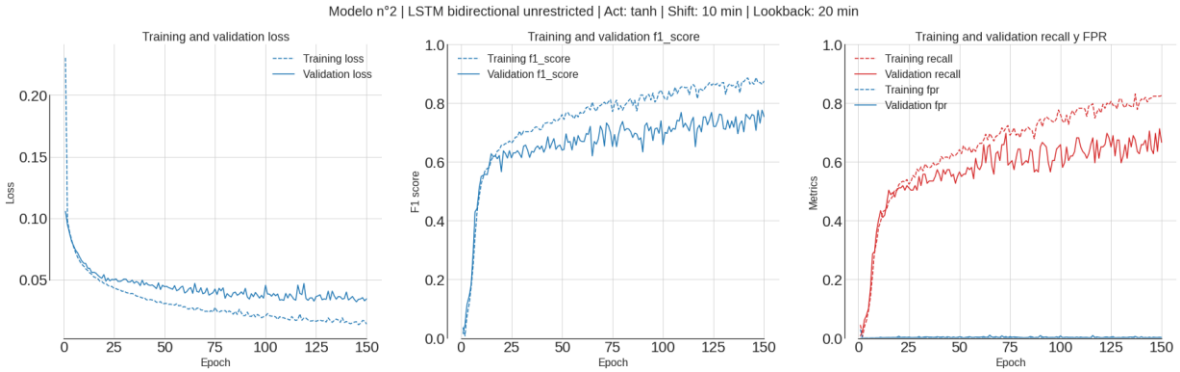


Modelo n°15 | LSTM 2 UNRESTRICTED STATELESS BIDIRECT | V.I | Act: tanh | Shift: 10 min | Lookback: 30 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0.5 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

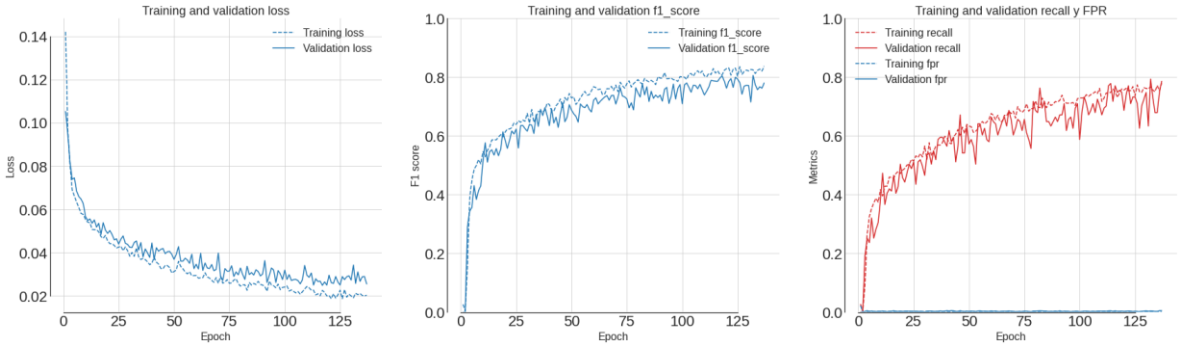


Modelos LSTM Versión I: Iteración III

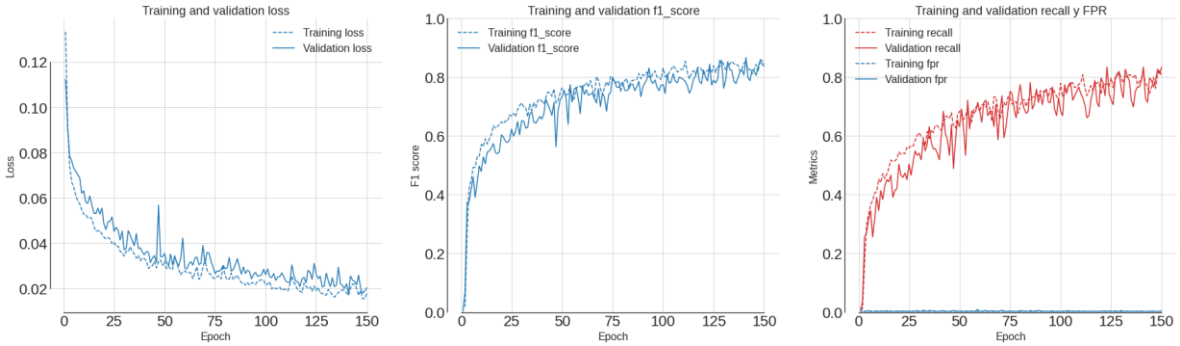




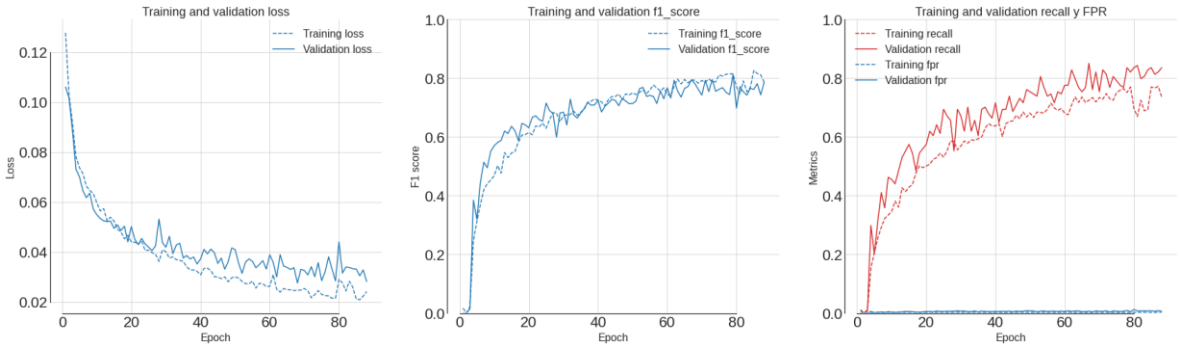
Modelo n°4 | LSTM bidirectional unrestricted | Act: tanh | Shift: 10 min | Lookback: 120 min



Modelo n°5 | LSTM bidirectional unrestricted | Act: tanh | Shift: 10 min | Lookback: 180 min

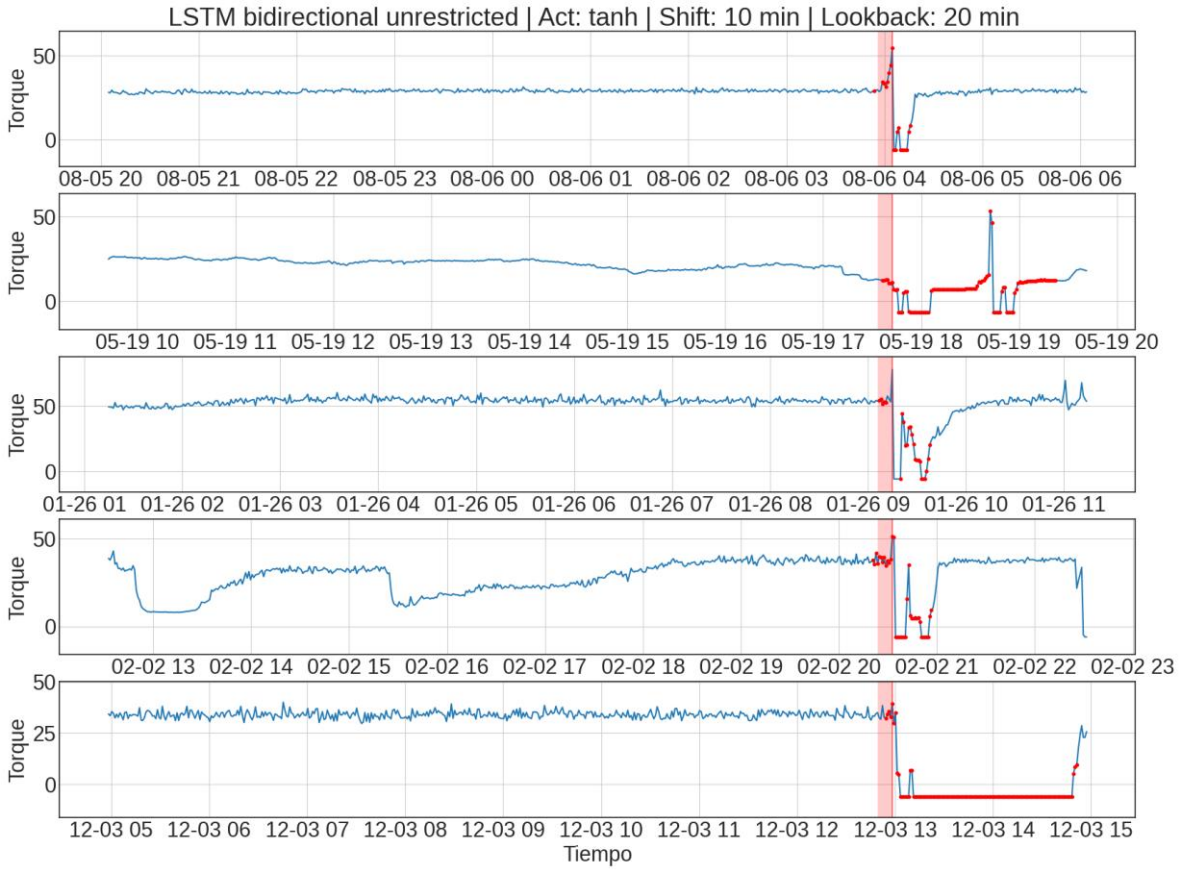


Modelo n°6 | LSTM bidirectional unrestricted | Act: tanh | Shift: 10 min | Lookback: 240 min

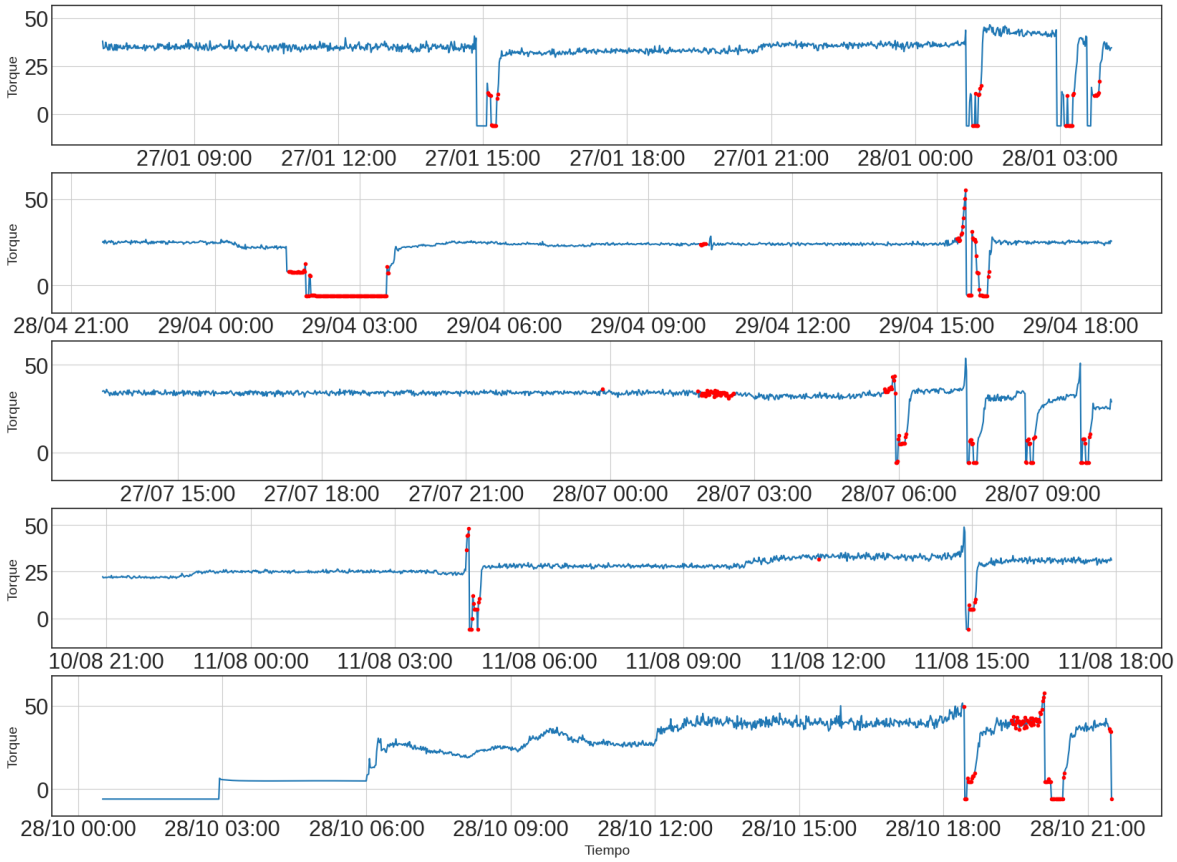


Modelo seleccionado

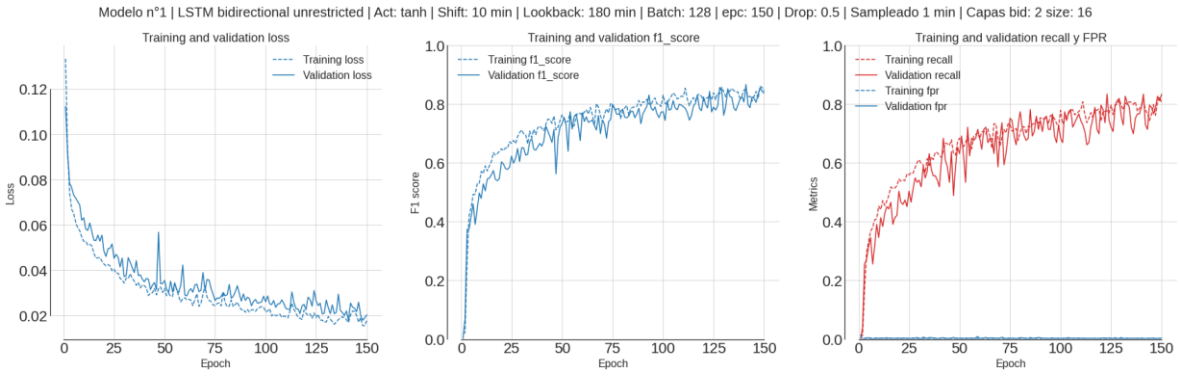
Eventos conocidos



Eventos no conocidos

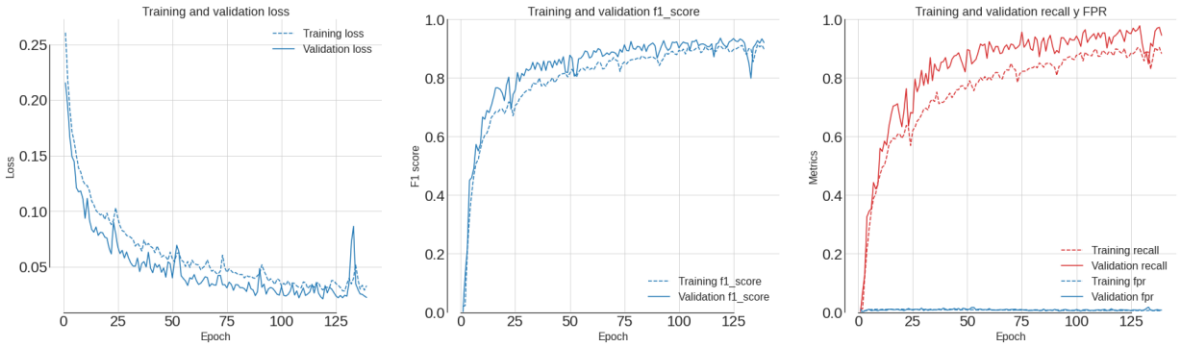


Curvas de aprendizaje modelos Iteración IV

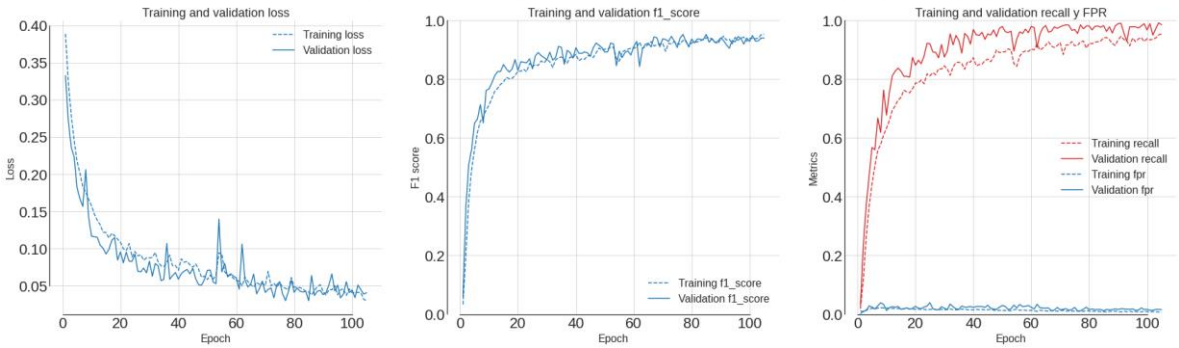


Modelo n°2 | LSTM bidirectional unrestricted | Act: tanh | Shift: 20 min | Lookback: 180 min | Batch: 128 | epc: 150 | Drop: 0.5 | Sampleado 1 min | Capas bid: 2 size: 16

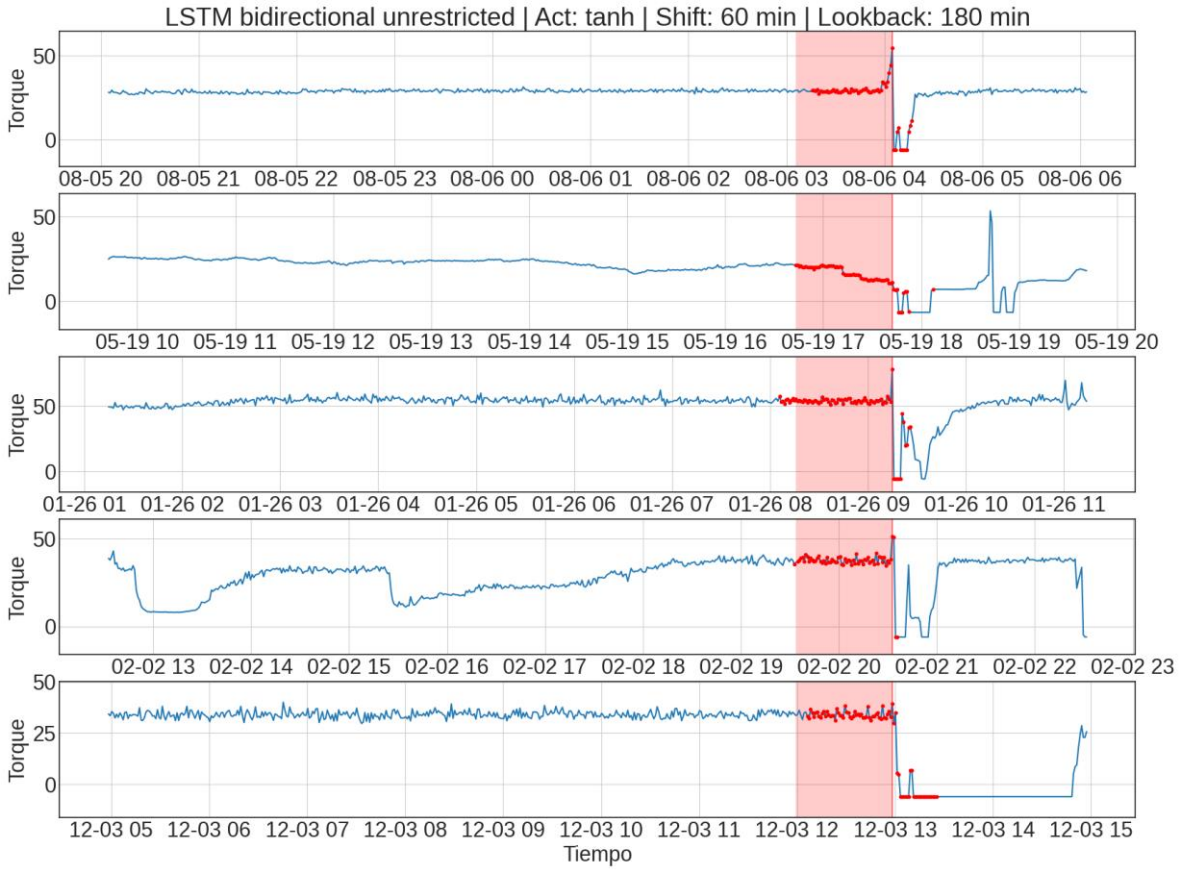
Modelo n°3 | LSTM bidirectional unrestricted | Act: tanh | Shift: 30 min | Lookback: 180 min | Batch: 128 | epc: 150 | Drop: 0.5 | Sampleado 1 min | Capas bid: 2 size: 16



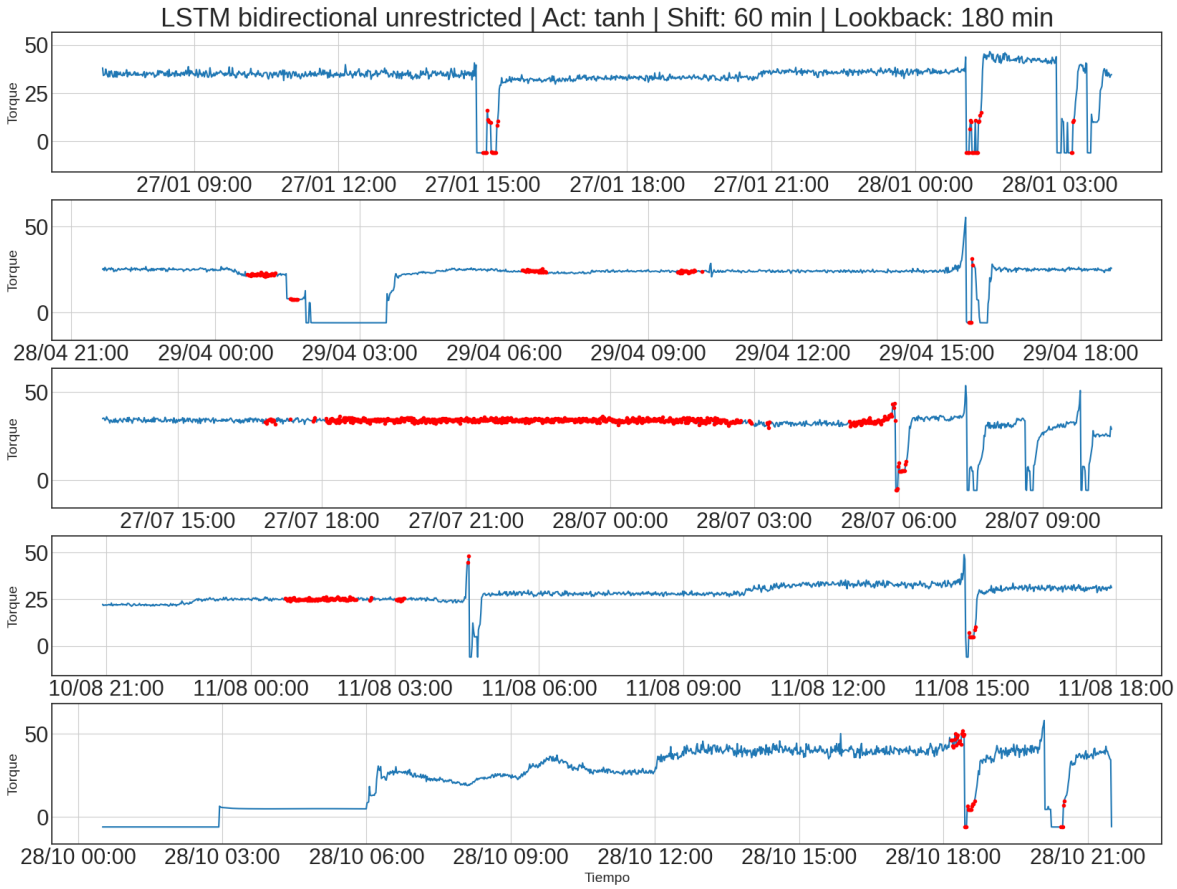
Modelo n°4 | LSTM bidirectional unrestricted | Act: tanh | Shift: 60 min | Lookback: 180 min | Batch: 128 | epc: 150 | Drop: 0.5 | Sampleado 1 min | Capas bid: 2 size: 16



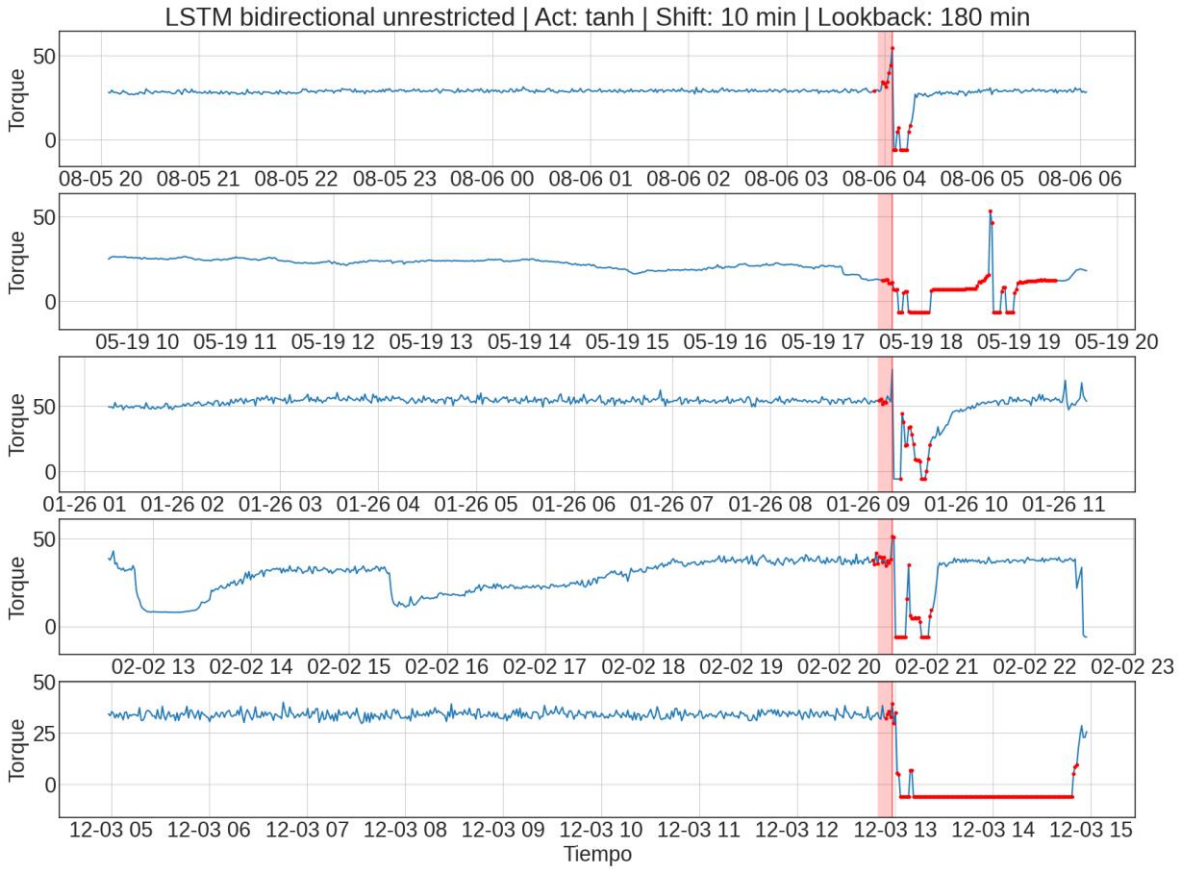
Eventos conocidos



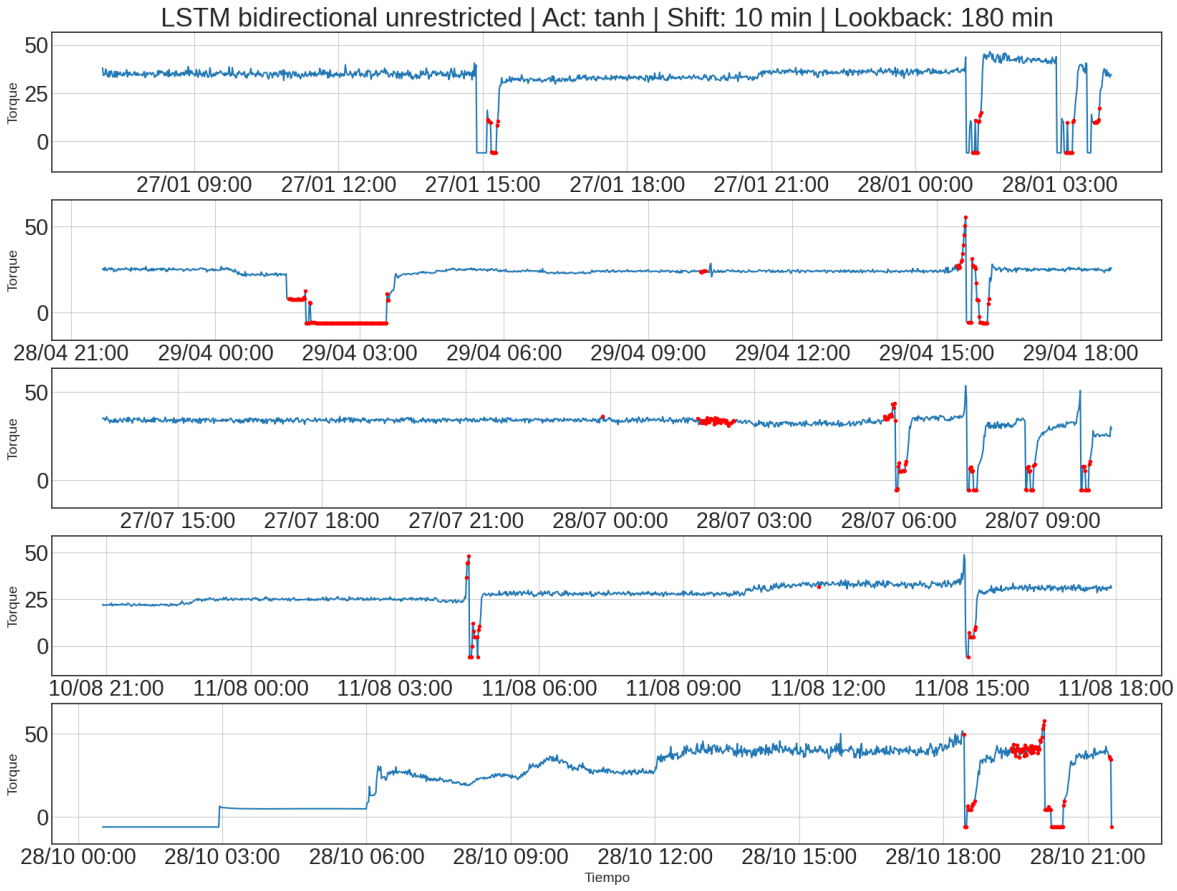
Eventos no conocidos



Eventos conocidos



Eventos no conocidos



Modelos CNN Versión I: Iteración I

Caso I

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_2 (Conv1D)            (None, 28, 9)             495

 max_pooling1d_2 (MaxPooling  (None, 9, 9)              0
 1D)

 flatten_2 (Flatten)          (None, 81)                0

 dense_2 (Dense)              (None, 8)                 656

 output (Dense)               (None, 1)                 9

=================================================================
Total params: 1,160
Trainable params: 1,160
Non-trainable params: 0
_________________________________________________________________
```

Caso II

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d (Conv1D)              (None, 28, 9)             495

 dropout (Dropout)            (None, 28, 9)             0

 max_pooling1d (MaxPooling1D  (None, 9, 9)              0
 )

 flatten (Flatten)            (None, 81)                0

 dense (Dense)                (None, 8)                 656

 output (Dense)               (None, 1)                 9

=================================================================
Total params: 1,160
Trainable params: 1,160
Non-trainable params: 0
_________________________________________________________________
```

Caso III

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 28, 9)             495

 dropout (Dropout)           (None, 28, 9)             0

 max_pooling1d (MaxPooling1D  (None, 9, 9)             0
 )

 flatten (Flatten)           (None, 81)                0

 dense (Dense)               (None, 8)                 656

 output (Dense)              (None, 1)                 9

=================================================================
Total params: 1,160
Trainable params: 1,160
Non-trainable params: 0
_________________________________________________________________
```

Caso IV

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 1, 9)          495

 max_pooling2d (MaxPooling2D  (None, 9, 1, 9)          0
 )

 flatten (Flatten)           (None, 81)                0

 dense (Dense)               (None, 8)                 656

 output (Dense)              (None, 1)                 9

=================================================================
Total params: 1,160
Trainable params: 1,160
Non-trainable params: 0
_________________________________________________________________
```

Caso V

Layer (type)	Output Shape	Param #
Conv2d (Conv2D)	(None, 28, 15, 9)	117
MaxPool (MaxPooling2D)	(None, 9, 3, 9)	0
Flatten (Flatten)	(None, 243)	0
Dense1 (Dense)	(None, 16)	3904
output (Dense)	(None, 1)	17

Total params: 4,038
Trainable params: 4,038
Non-trainable params: 0

Caso VI

Layer (type)	Output Shape	Param #
Convlayer (Conv1D)	(None, 27, 18)	1314
dropout (Dropout)	(None, 27, 18)	0
Conv1x1 (Conv1D)	(None, 27, 9)	171
maxpooling (MaxPooling1D)	(None, 9, 9)	0
flatten (Flatten)	(None, 81)	0
dense (Dense)	(None, 8)	656
output (Dense)	(None, 1)	9

Total params: 2,150
Trainable params: 2,150
Non-trainable params: 0

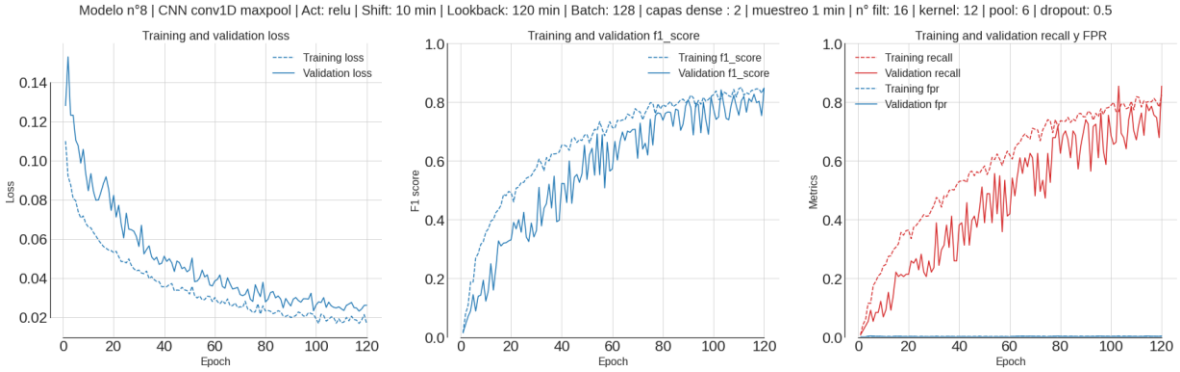
Modelos CNN Versión I: Iteración II

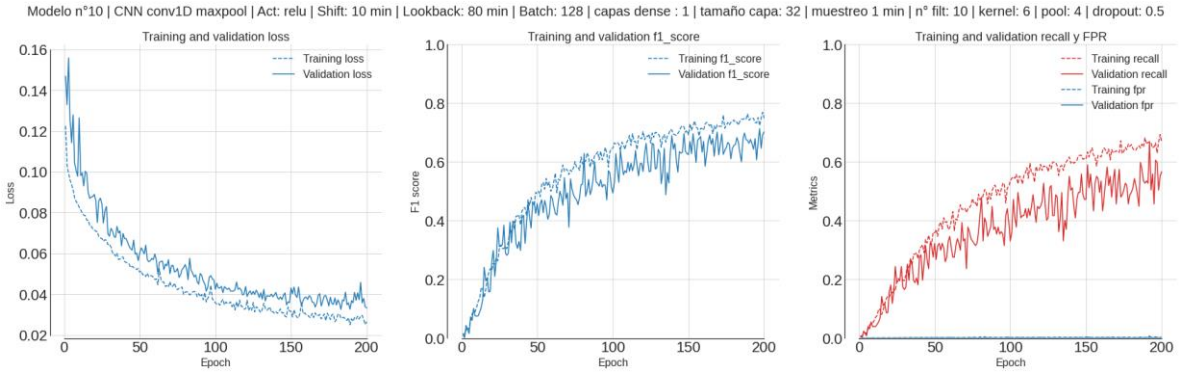
n°	Activacion	Capas Conv (Filtros/Kernel)	Capas MaxPool (Pool)	Dropout	Capas Densas	Etiquetas positivas [min]	Tamaño Ventana [min]
1	relu	Conv1d (16/12)	(8)	0.0	(32)(16)	10	15
2	relu	Conv1d (9/4)	(3)	0.0	(64)(32)(16)	10	60
3	relu	Conv1d (9/4)	(3)	0.0	(64)(32)(16)	10	60
4	relu	Conv1d (16/12)	(6)	0.0	(64)(32)(16)	10	60
5	relu	(16/12)	(6)	0.0	(16)	10	60
6	relu	(16/4)	(4)	0.5	(32)	10	60
7	tanh	(16/4)	(4)	0.5	(32)	10	60
8	relu	(16/12)	(6)	0.5	(16)	10	120
9	relu	(10/6)	(4)	0.0	(32)	10	80
10	relu	(10/6)	(4)	0.5	(32)	10	80
11	relu	(10/15)	(4)	0.5	(32)	10	80
12	relu	(32/15)	(4)	0.5	(128)(32)	10	80
13	relu	(16/20)	(4)	0.5	(32)	10	80
14	tanh	(12/3)	(2)	0.0	(16)	10	15
15	tanh	(12/3)	(2)	0.5	(16)	10	15
16	tanh	(12/10)	(2)	0.0	(16)	10	15
17	tanh	(12/10)	(1)	0.0	(64)(16)	10	60
18	tanh	(9/10)	(1)	0.0	(64)(16)	10	60
19	tanh	(9/5)	(1)	0.0	(64)(16)	10	60

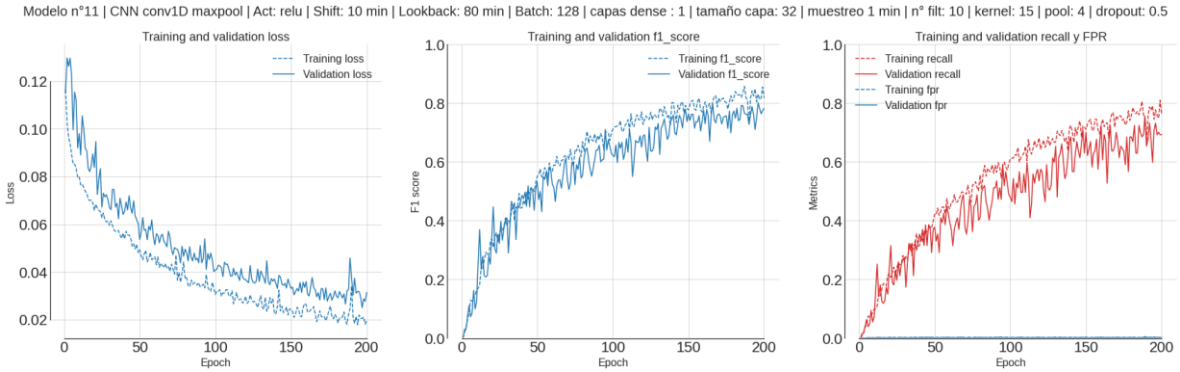
20	tanh	(9/10)	(4)	0.5	(64)(16)	10	60
21	tanh	(18/10)	(4)	0.0	(64)(16)	10	60
22	tanh	(9/4)	(3)	0.0	(64)(16)	10	60
23	relu	Conv2D (9/(4,1))	2D (3,1)	0.0	(64)	10	60
24	tanh	Conv2D (9/(4,1))	(3,1)	0.0	(64)	10	60
25	relu	Conv2D neigh (9/(4,4))	2D (3,4)	0.0	(64)(16)	10	60
26	tanh	Conv1D (32/4)(16/4)	1D (2)	0.0	(64/16)	10	60
27	tanh	Conv1D (32/4)(16/4)	1D (2)	0.5	(64/16)	10	60
28	tanh	Conv1D (16/5)(8/5)	1D (1)	0.0	(64/16)	10	60
29	relu	Conv1D (16/10)(8/10)	1D (1)	0.0	(64)(16)	10	60
30	relu	Conv1D (16/8)(16/4)	1D (3)(1)	0.0	(16)	10	60
31	relu	Conv1D (8/8)(8/4)(8/2)	1D (3)(1)	0.0	(16)	10	60
32	tanh	Conv1D (8/8)(8/4)(8/2)	1D (3)(1)	0.0	(16)	10	60
33	tanh	Conv1D (12/4)(4/12)	1D (1)(1)	0.0	(16)	10	60
34	tanh	Conv1D (12/4)(6/12)	1D (2)(2)	0.0	(8)	10	60
35	tanh	Conv1D (12/4)(6/12)	1D (1)(2)	0.0	(16)	10	60
36	tanh	Conv1D	NO	0.0	(16)	10	60

		(12/4)(4/20)					
37	tanh	Conv1D (12/4)(6/16)(4/32)	NO	0.0	(8)	10	60
38	tanh	Conv1D (12/4)(6/16)(4/32)	NO	0.5	(8)	10	60
39	tanh	Conv1D (12/4)(6/16)	1D (4)	0.0	(8)	10	60
40	tanh	Conv1D (12/4)(6/16)	1D (1)(2)	0.0	(8)	10	60
41	relu	Conv1D (9/3)	1D (3)	0.0	(8)	10	30
42	tanh	Conv1D (9/3)	1D (3)	0.5	(8)	10	30
43	relu	Conv1D (9/3)	1D (3)	0.5	(8)	10	30
44	relu	Conv2D (9/(3,1))	1D (3)	0.0	(8)	10	30
45	relu	Conv2D neigh (9/(3,4))	1D (3,4)	0.0	(8)	10	30
46	relu	Conv1D (18/3)(9/1)	1D (3)	0.0	(8)	10	30
47	tanh	Conv1D (9/3)	1D (3)	0.0	(8)	10	30

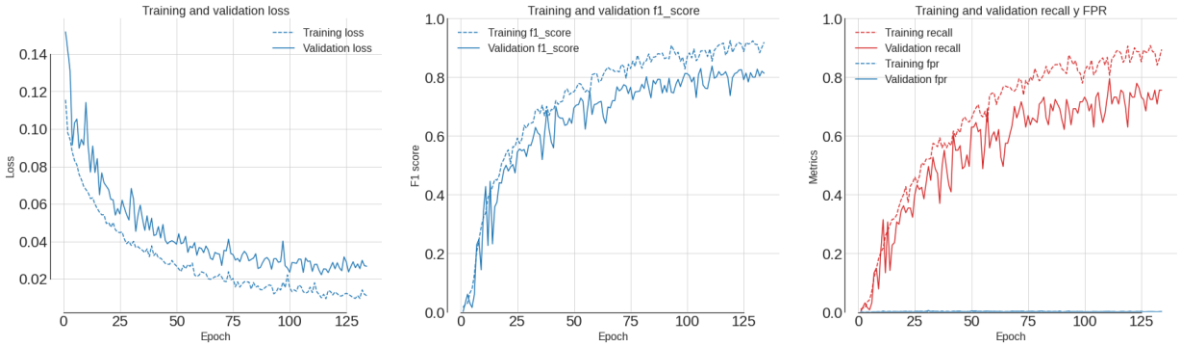
Curvas de aprendizaje modelos CNN: Iteración II



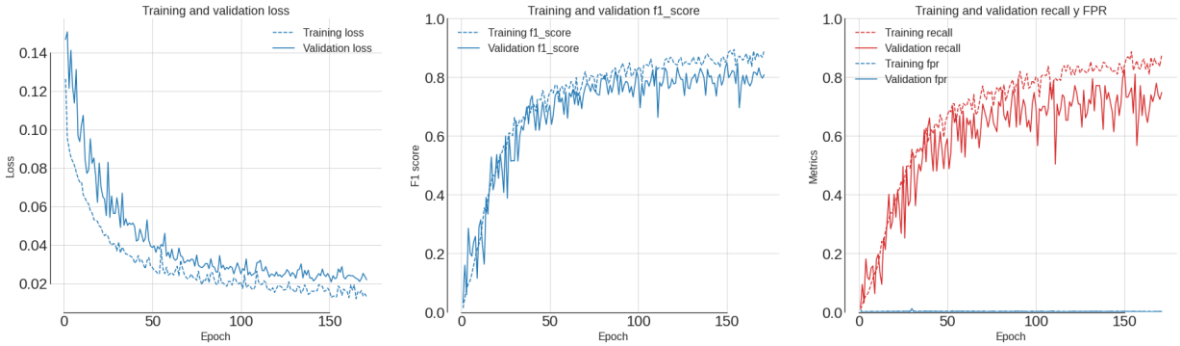




Modelo n°12 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 80 min | Batch: 128 | capas dense : 2 | tamaño capa: 128 | muestreo 1 min | n° filt: 32 | kernel: 15 | pool: 4 | dropout: 0.5

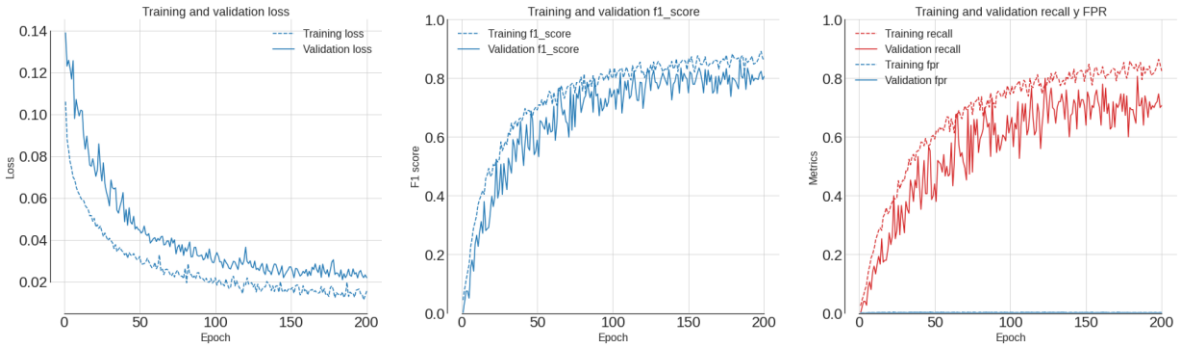


Modelo n°13 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 80 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

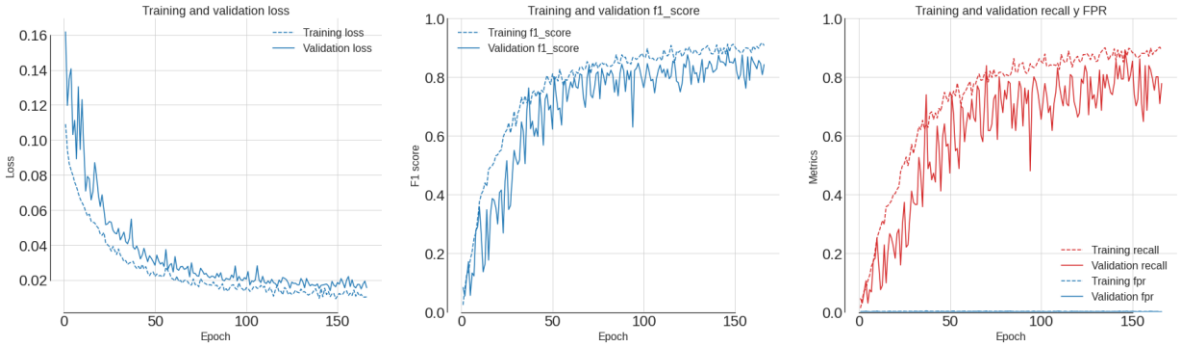


Modelos CNN Versión I: Iteración III

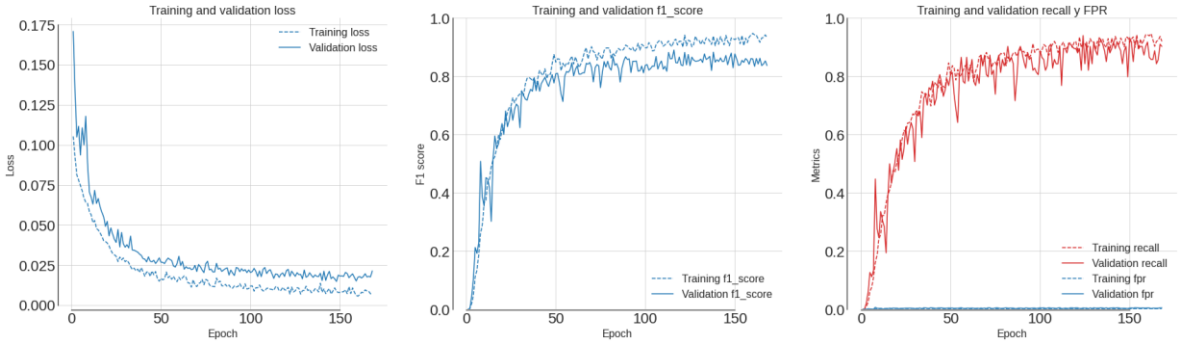
Modelo n°1 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 60 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5



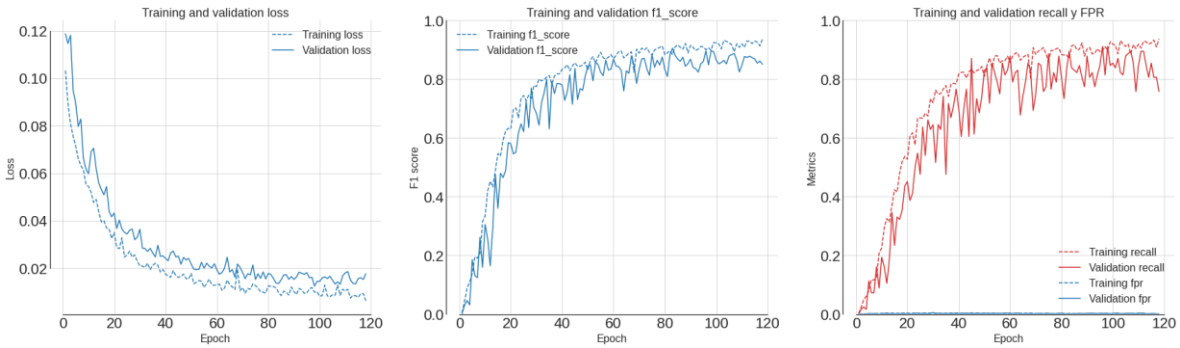
Modelo n°2 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 120 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5



Modelo n°3 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

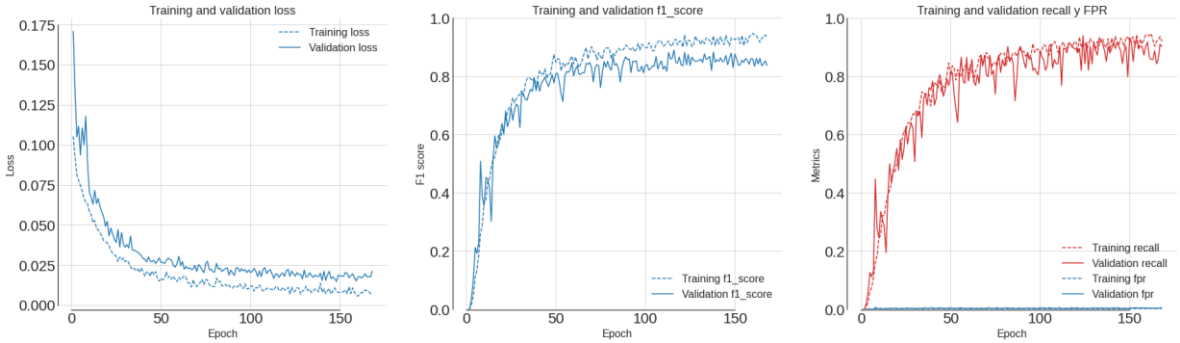


Modelo n°4 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 360 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

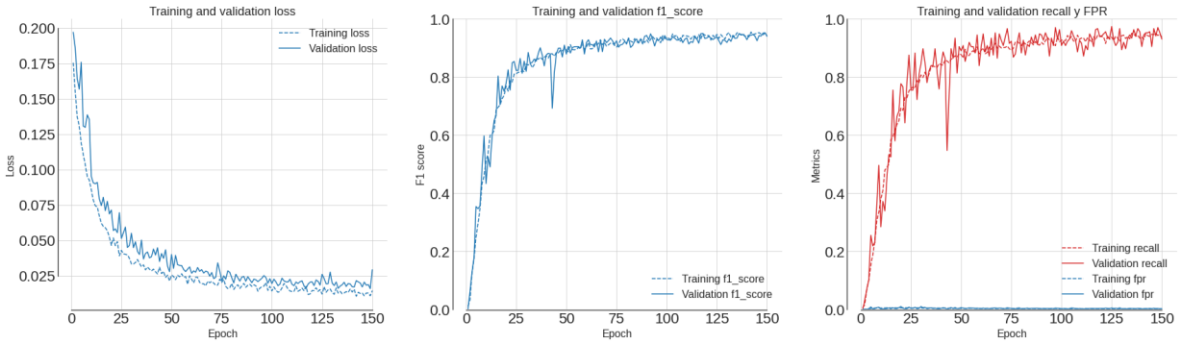


Modelos CNN Versión I: Iteración IV

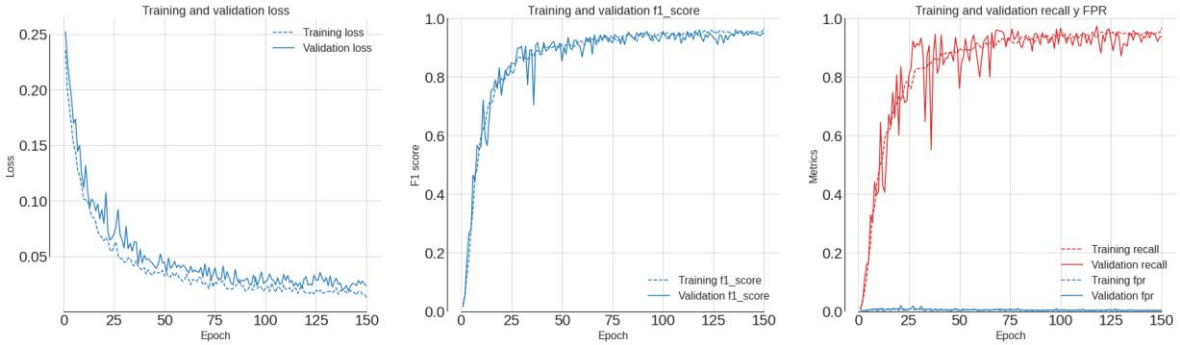
Modelo n°1 | CNN conv1D maxpool | Act: relu | Shift: 10 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

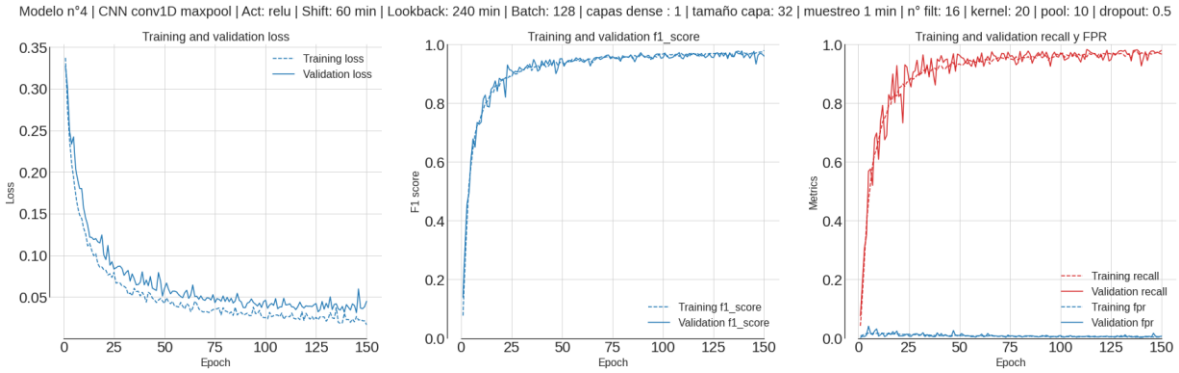


Modelo n°2 | CNN conv1D maxpool | Act: relu | Shift: 20 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

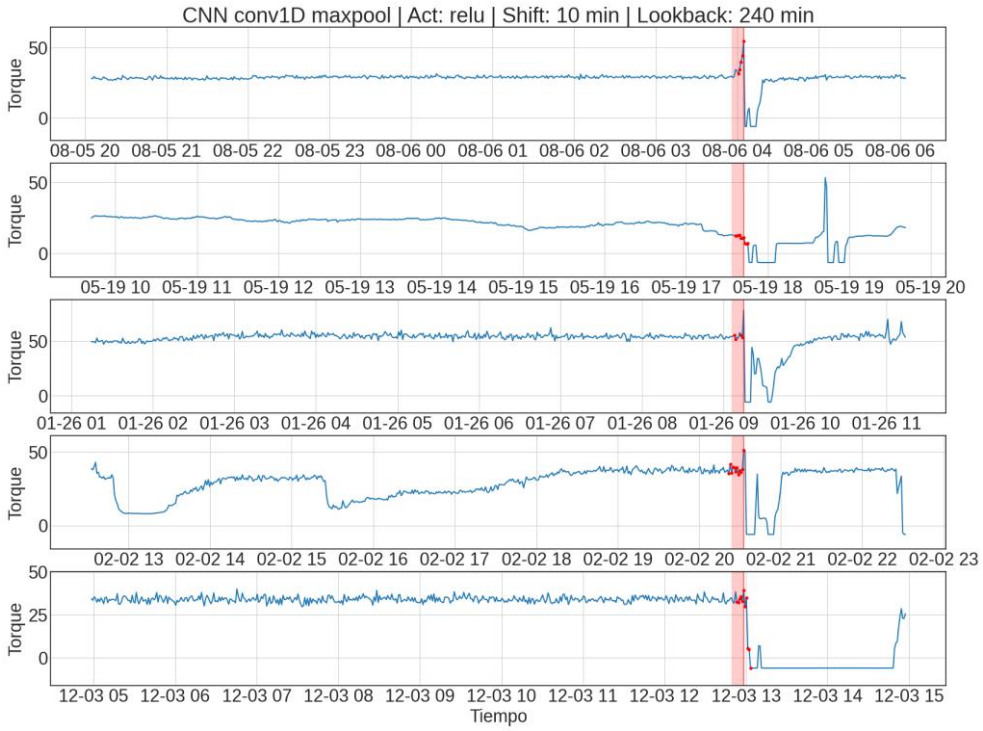


Modelo n°3 | CNN conv1D maxpool | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 32 | muestreo 1 min | n° filt: 16 | kernel: 20 | pool: 10 | dropout: 0.5

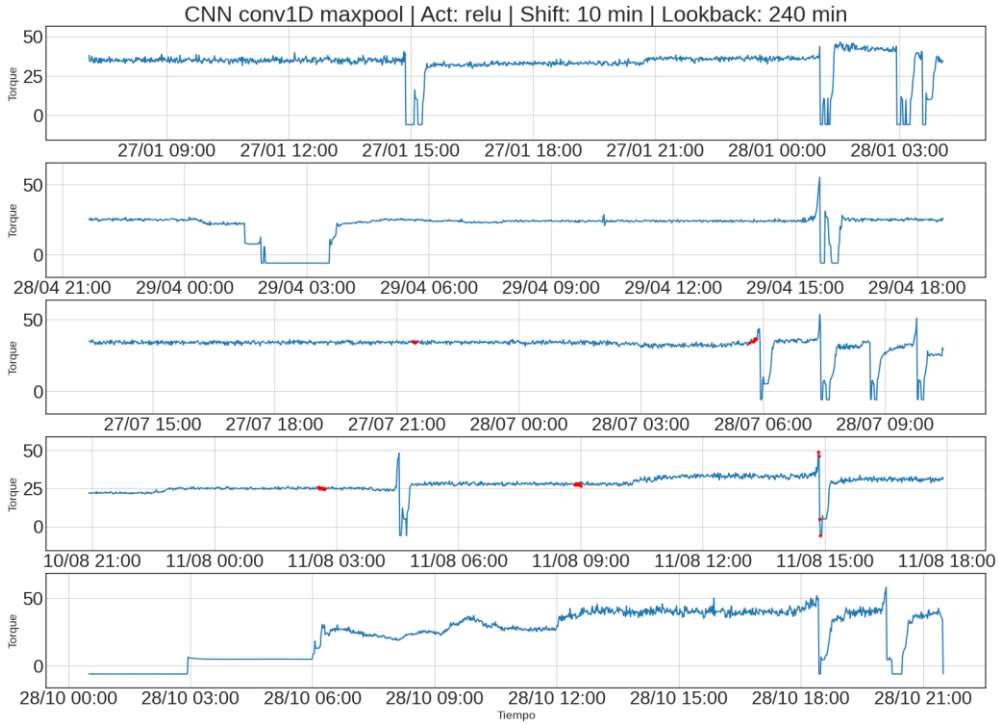




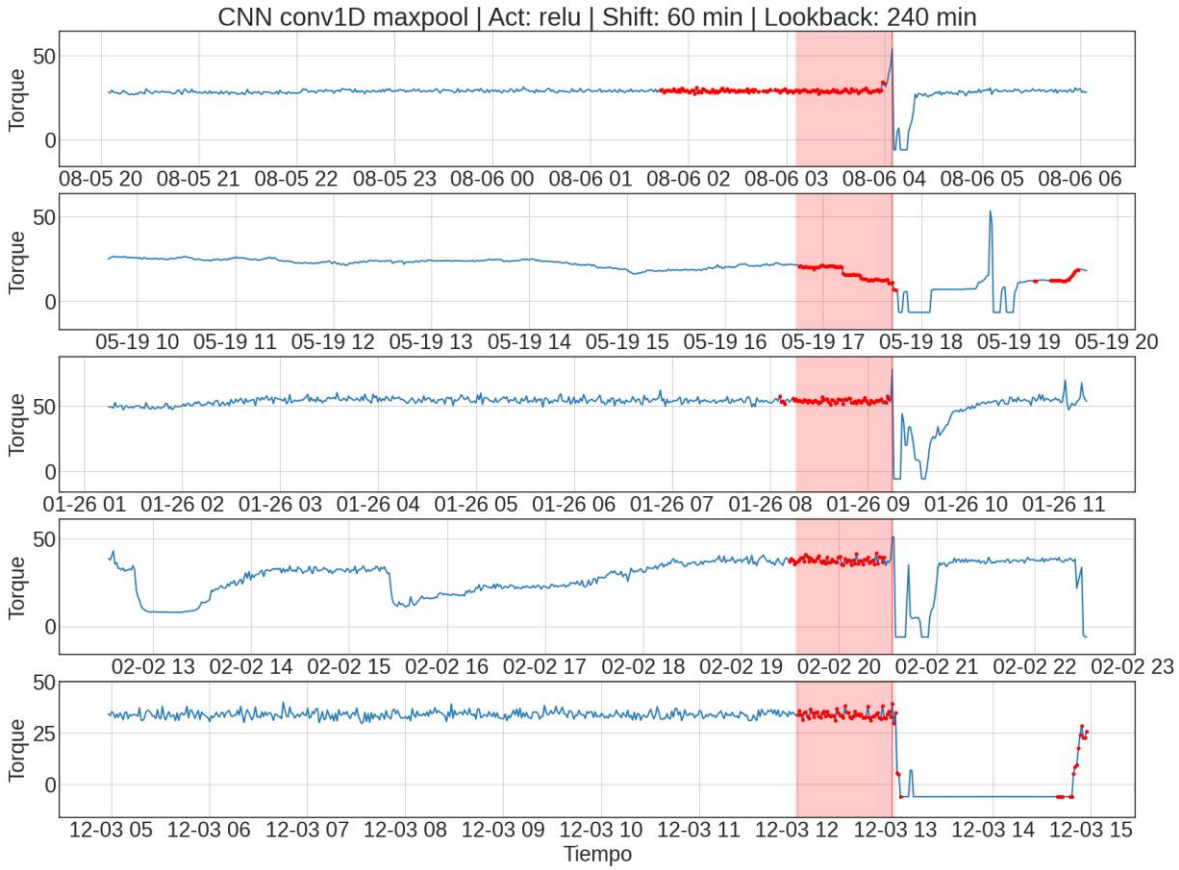
Eventos conocidos



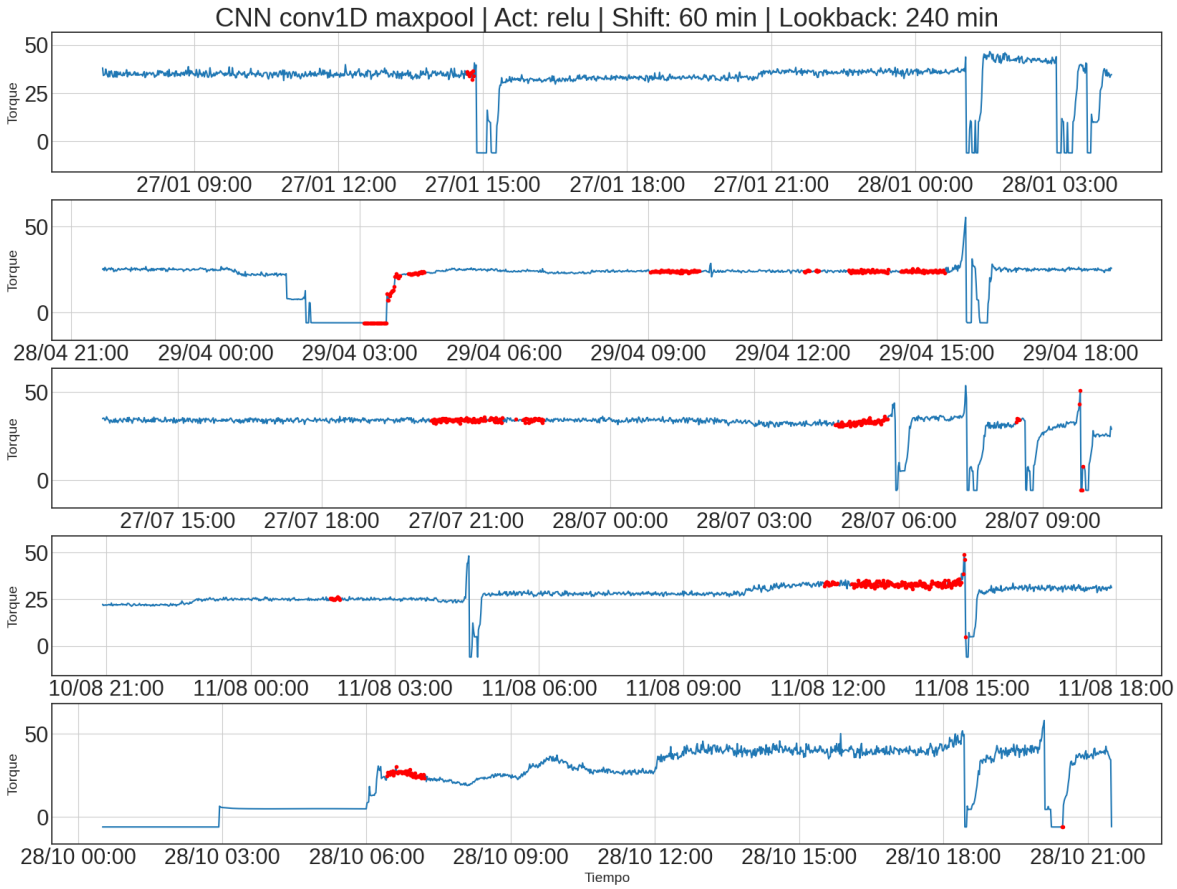
Eventos no conocidos



Eventos conocidos



Eventos no conocidos

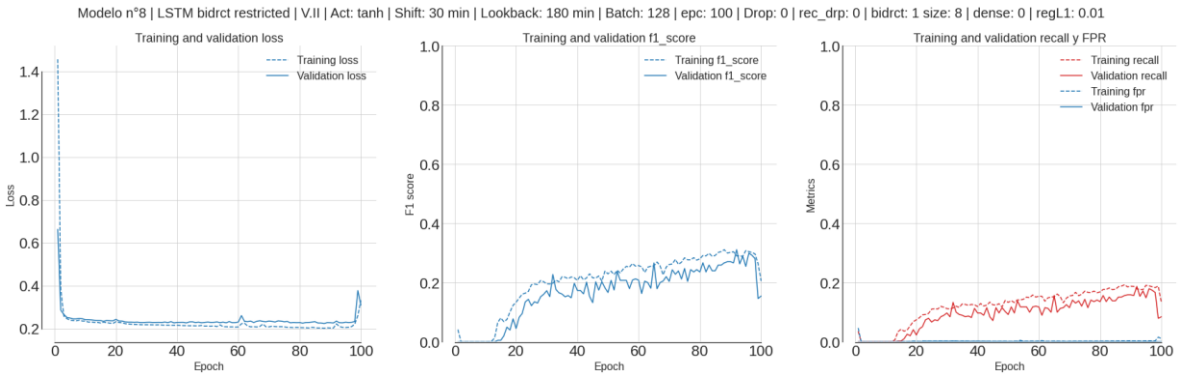


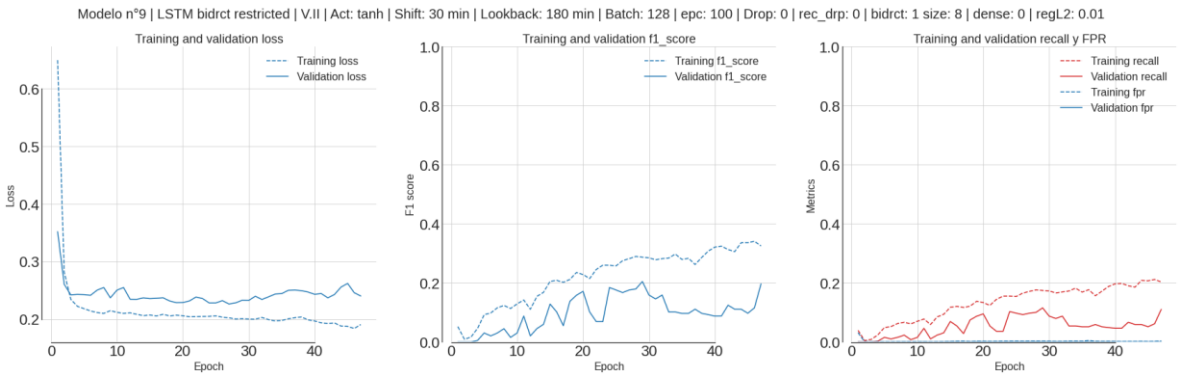
Modelos LSTM Versión II

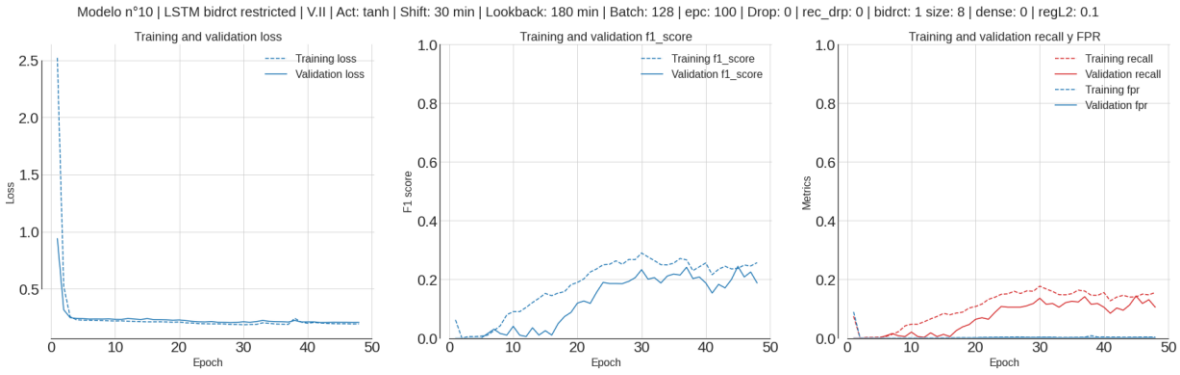
n°	Act	Dropout / Drop_Rec	Capas LSTM	Capas Densas	Estructura	Batch	Reg L1/L2
1	tanh	0.5/0.0	(16)(8)	NO	Unrestricted Stateless Bidirectional	128	0.0/0.0
2	tanh	0.0/0.0	(8)(8)	(32)	Unrestricted Stateless Bidirectional	128	0.0/0.0
3	tanh	0.5/0.0	(8)(4)	NO	Unrestricted Stateless Bidirectional	128	0.0/0.0
4	tanh	0.5/0.0	(16)(8)(4)	NO	Unrestricted Stateless Bidirectional	128	0.0/0.0
5	tanh	0.2/0.0	(8)(4)	NO	Unrestricted Stateless Bidirectional	128	0.0/0.0
6	tanh	0.0/0.0	(8)(4)	NO	Unrestricted Stateless Bidirectional	128	0.0/0.0
7	tanh	0.0/0.0	(16)	NO	Restricted Stateless Bidirectional	128	0.0/0.0
8	tanh	0.0/0.0	(16)	NO	Restricted Stateless Bidirectional	128	0.01/0.0
9	tanh	0.0/0.0	(16)	NO	Restricted Stateless Bidirectional	128	0.0/0.01
10	tanh	0.0/0.0	(16)	NO	Restricted Stateless Bidirectional	128	0.0/0.1

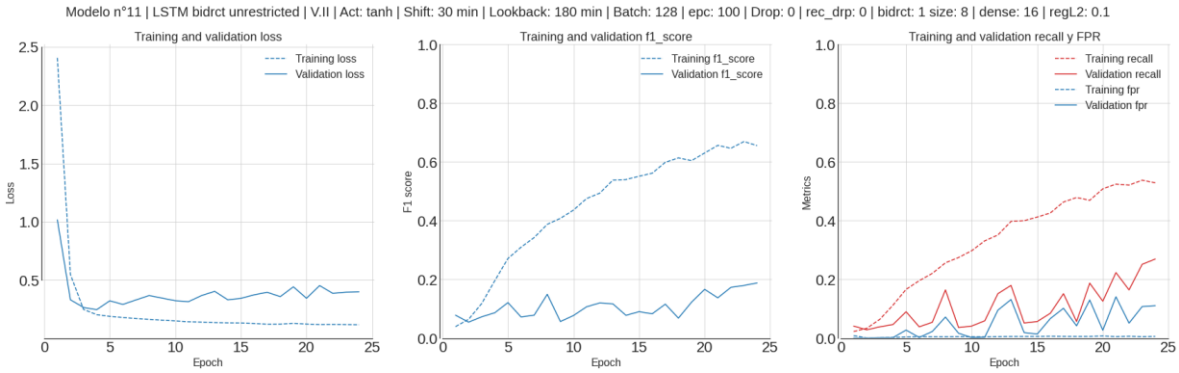
11	tanh	0.0/0.0	(8)	(16)	Unrestricted Stateless Bidirectional	128	0.0/0.1
12	tanh	0.0/0.0	(8)(4)	NO	Unrestricted Stateless Bidirectional	32	0.1/0.0

Curvas de aprendizaje mejores modelos









Modelos n°8, n°9 y n°10 con la misma arquitectura

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 16)	1728
flatten (Flatten)	(None, 16)	0
output (Dense)	(None, 1)	17

Total params: 1,745
Trainable params: 1,745
Non-trainable params: 0

Arquitectura modelo n°11

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 180, 16)	1728
flatten (Flatten)	(None, 2880)	0
dense (Dense)	(None, 16)	46096
output (Dense)	(None, 1)	17

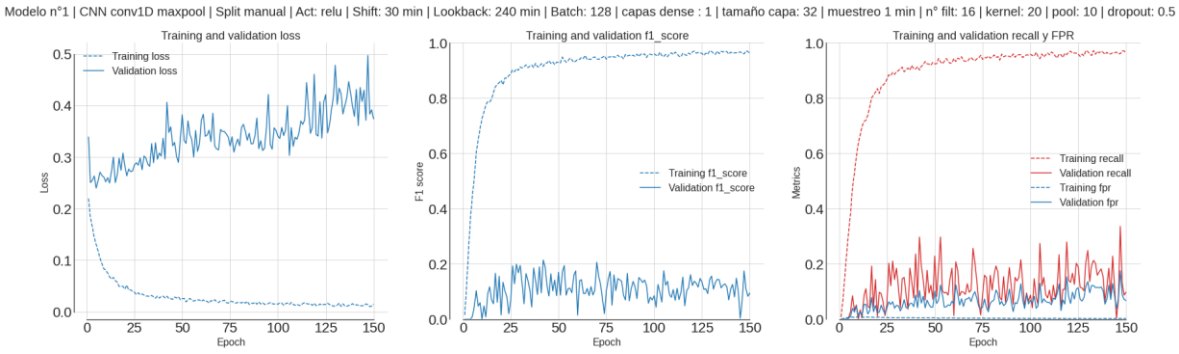
Total params: 47,841
Trainable params: 47,841
Non-trainable params: 0

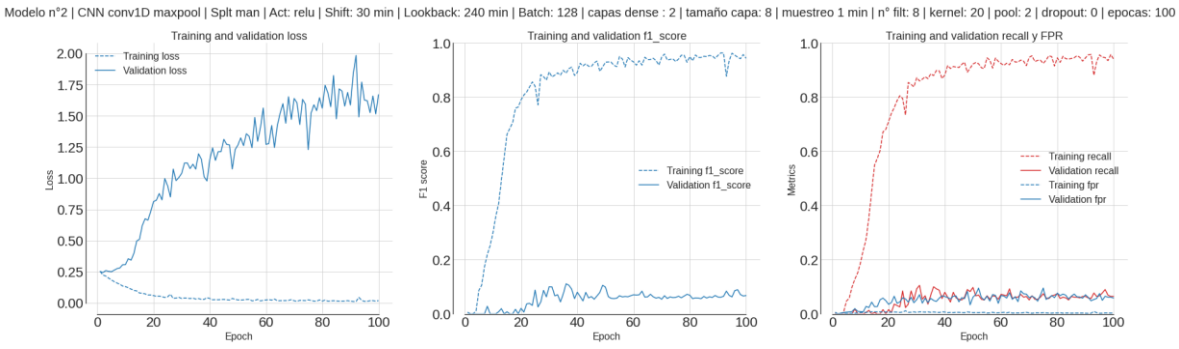
Modelos CNN Versión II

n°	Capas Conv (Filtros/Kernel)	Capas MaxPool (Pool)	Drop out	Capas Densas	Reg L1/L2	Opt/LR	Batch
1	Conv1d (16/20)	1D (10)	0.5	(32)	0.0/0.0	Adam/0.001	128
2	Conv1d (8/20)(4/80)	1D (2)(4)	0.0	(8)	0.0/0.0	Adam/0.001	128
3	Conv1d (8/20)(4/80)	1D (2)(4)	0.5	(8)	0.0/0.0	Adam/0.001	128
4	Conv1d (8/20)	1D (10)	0.5	(16)	0.0/0.0	Adam/0.001	128
5	Conv1d (32/10)(16/20)(10/40)	1D (1)(2)(4)	0.5	(16)	0.0/0.0	Adam/0.001	128
6	Conv1d (16/20)	1D (10)	0.2	(32)	0.0/0.0	Adam/0.001	128
7	Conv1d (16/20)(8/40)(4/80)	1D (10)(20)(40)	0.0	(32)	0.0/0.0	Adam/0.001	128
8	Conv1d (8/20)(4/80)	1D (2)(4)	0.1	(8)	0.0/0.0	Adam/0.001	128
9	Conv1d (8/20)	1D (10)	0.0	(8)	0.0/0.0	Adam/0.001	128
10	Conv1d (8/8)	1D (3)	0.0	(16)	0.0/0.0	Adam/0.001	128
11	Conv1d (8/20)	NO	0.0	(16)	0.0/0.0	Adam/0.001	128
12	Conv1d (8/20)	1D (1)	0.1	(16)	0.0/0.0	Adam/0.001	128
13	Conv1d (8/20)(4/80)	1D (2)(2)	0.0	(8)	0.0/0.0	AdaGrad/0.0001	128
14	Conv1d	1D	0.0	(8)	0.0/0.0	AdaDelta/0.00001	128

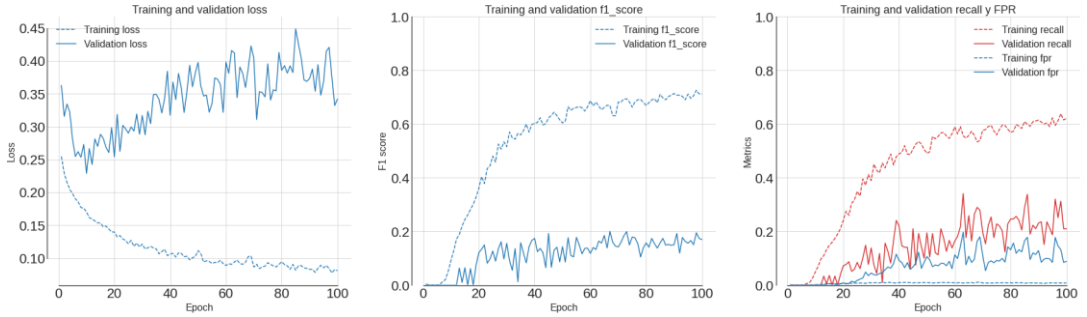
	(8/20)(4/80)	(2)(2)					
15	Conv1d (8/20)(4/80)	1D (2)(4)	0.1	(8)	0.1/0.0	Adam/0.001	128
16	Conv1d (32/20)(16/80)	1D (2)(4)	0.2	(8)	0.0/0.0	Adam/0.001	128
17	Conv1d (16/20)(8/80)	1D (2)(4)	0.0	(8)	0.0/0.0	Adam/0.001	128
18	Conv1d (16/10)(8/40)	1D (2)(4)	0.2	(8)	0.1/0.0	Adam/0.001	64
19	Conv1d (8/20)(4/80)	1D (2)(4)	0.5	(8)	0.1/0.0	Adam/0.001	64

Curvas de aprendizaje

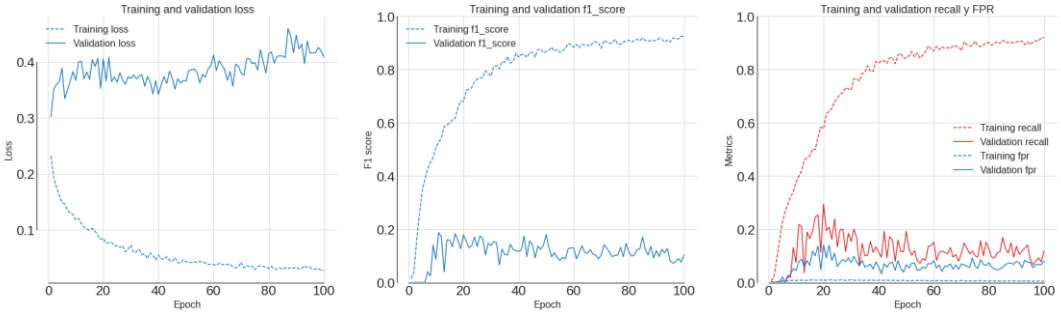




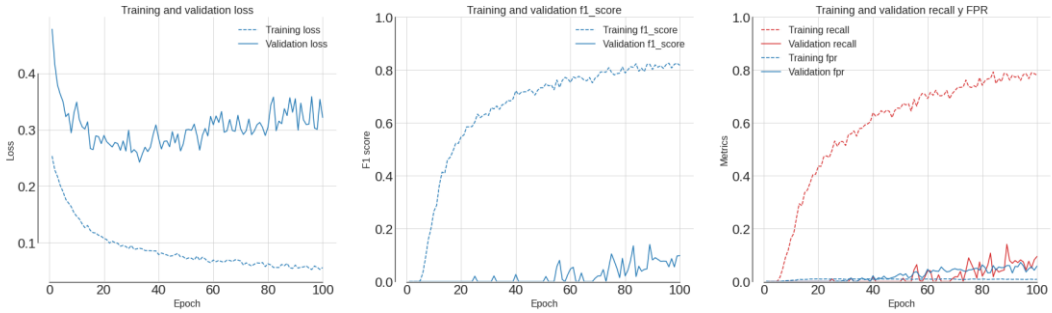
Modelo n°3 | CNN conv1D maxpool | Splt man | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | capas dense : 2 | tamaño capa: 8 | muestreo 1 min | n° filt: 8 | kernel: 20 | pool: 2 | dropout: 0.5 | epocas: 100



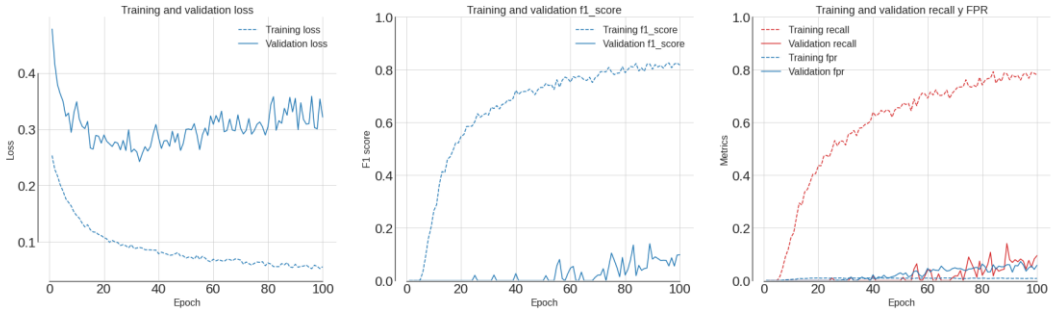
Modelo n°4 | CNN conv1D maxpool | Splt man | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 16 | muestreo 1 min | n° filt: 8 | kernel: 20 | pool: 10 | dropout: 0.5 | epocas: 100



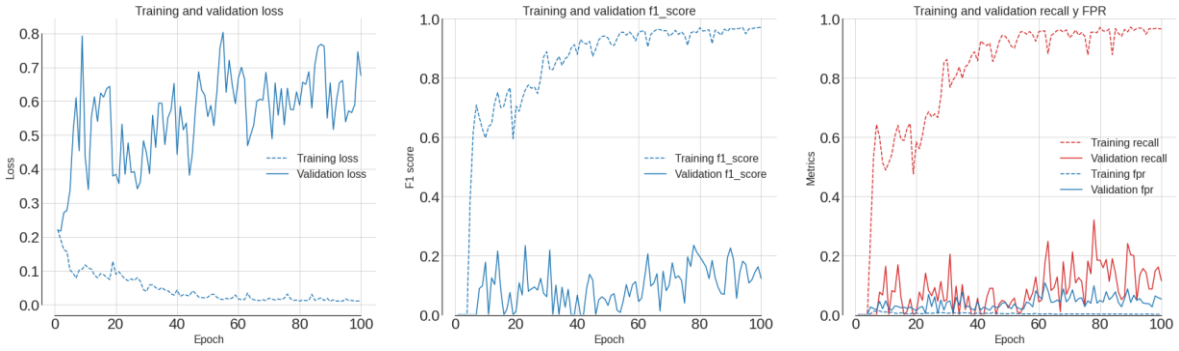
Modelo n°5 | CNN 3 conv1D maxpool | Splt man | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 16 | muestreo 1 min | n° filt: 32 | kernel: 10 | pool: 1 | dropout: 0.5 | epocas: 100



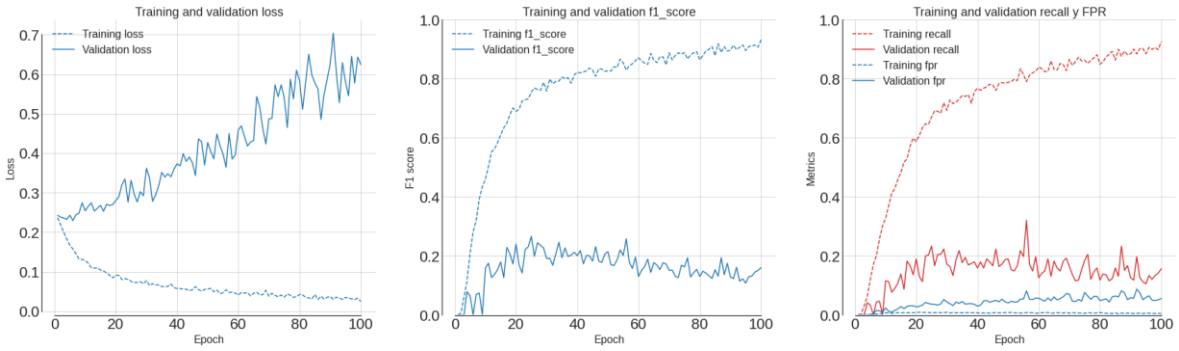
Modelo n°5 | CNN 3 conv1D maxpool | Splt man | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | capas dense : 1 | tamaño capa: 16 | muestreo 1 min | n° filt: 32 | kernel: 10 | pool: 1 | dropout: 0.5 | epocas: 100



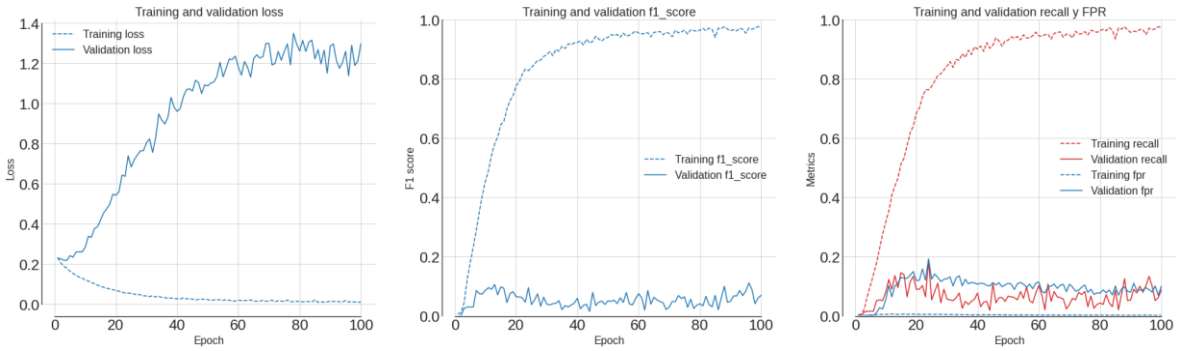
Modelo n°7 | CNN 3 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 32 | n° filt: 16 | kernel: 20 | pool: 10 | str_pool: 1 | drop: 0 | epc: 100



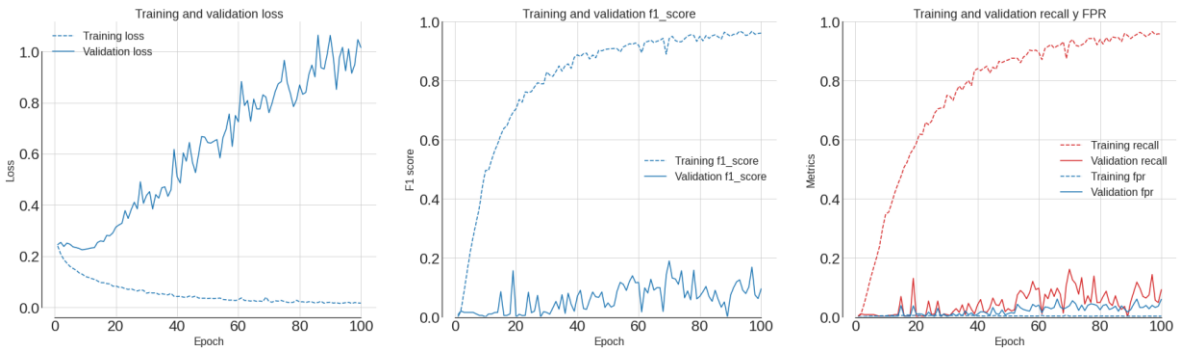
Modelo n°8 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.1 | epc: 100



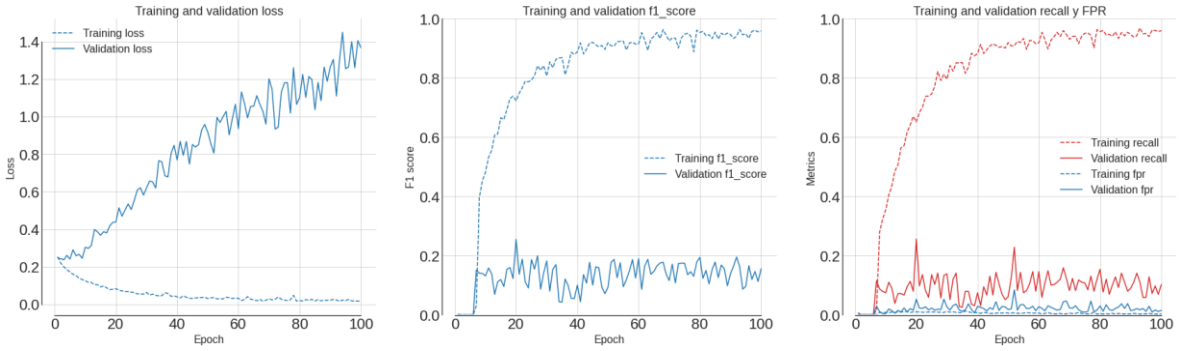
Modelo n°9 | CNN 1 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 10 | str_pool: 10 | drop: 0 | epc: 100



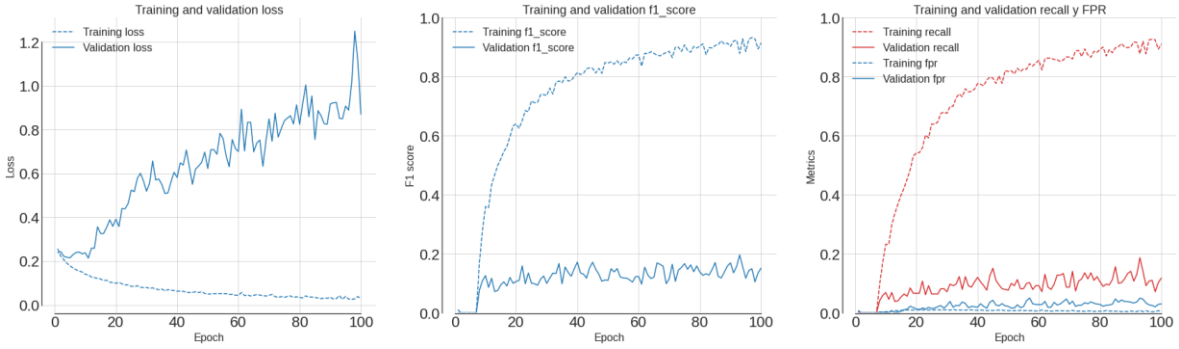
Modelo n°10 | CNN 1 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 16 | n° filt: 8 | kernel: 8 | pool: 3 | str_pool: 3 | drop: 0 | epc: 100



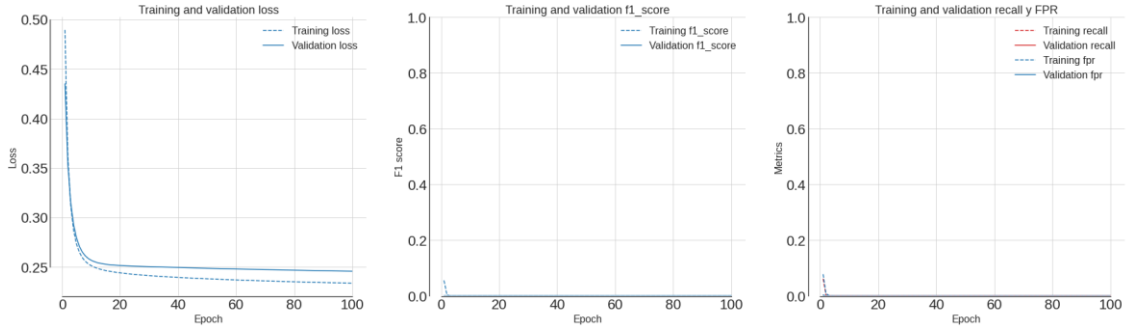
Modelo n°11 | CNN 1 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 16 | n° filt: 8 | kernel: 20 | pool: 0 | str_pool: 0 | drop: 0 | epc: 100



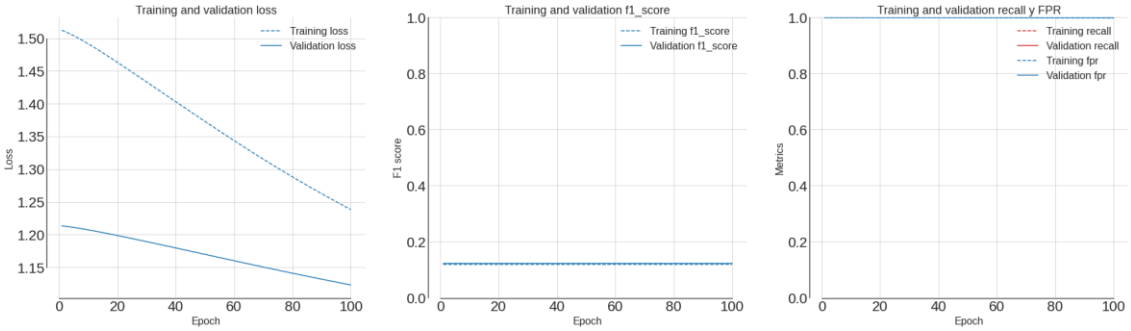
Modelo n°12 | CNN 1 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 16 | n° filt: 8 | kernel: 20 | pool: 1 | str_pool: 1 | drop: 0.1 | epc: 100



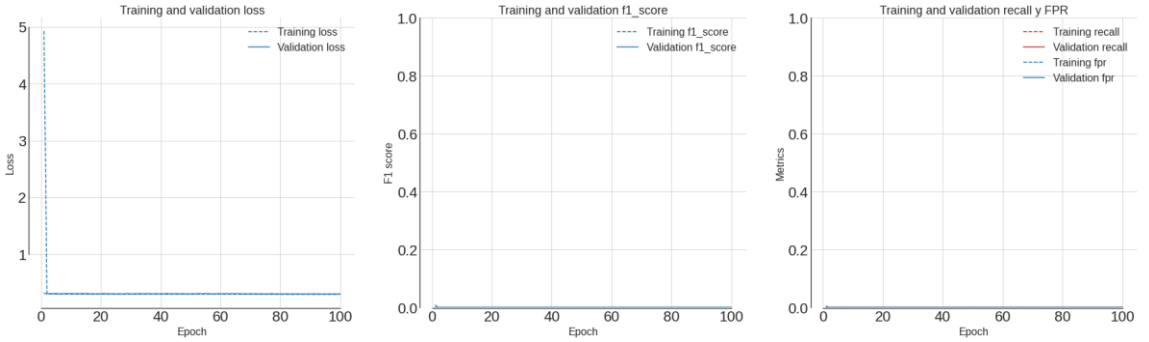
Modelo n°13 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.1 | epc: 100 | AdaGrad:0.0001



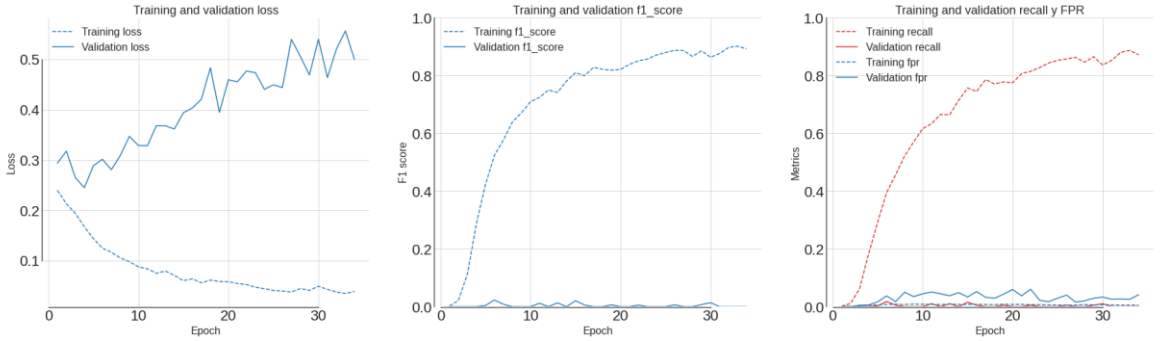
Modelo n°14 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.1 | epc: 100 | Adadelta:1e-05



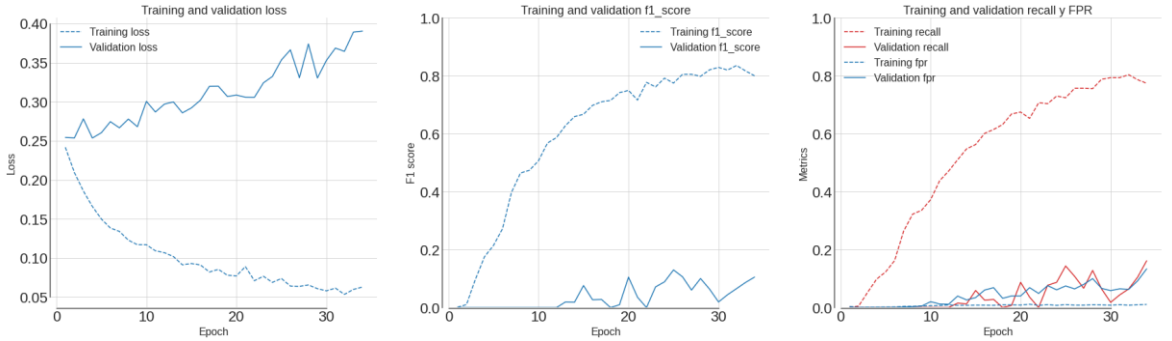
Modelo n°15 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.1 | epc: 100 | regL1: 0.1



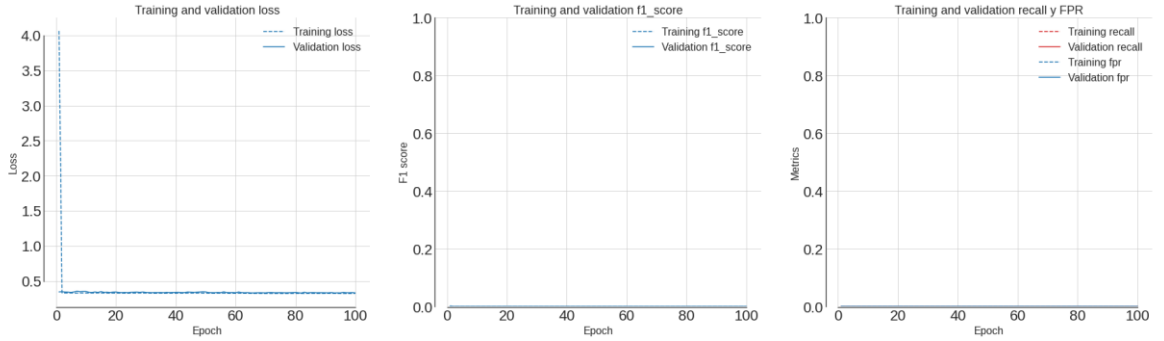
Modelo n°16 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 32 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.2 | epc: 100 | regL1: 0



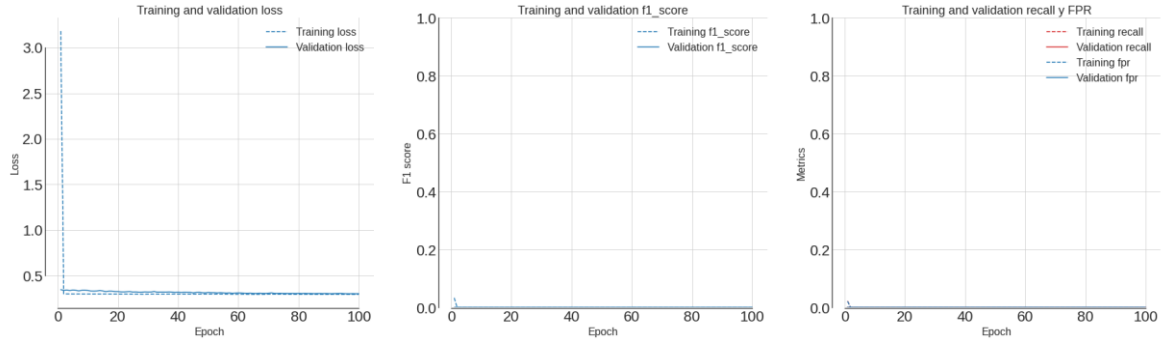
Modelo n°17 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 128 | dense : 1 | tamaño: 8 | n° filt: 16 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.2 | epc: 100 | regL1: 0



Modelo n°18 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 64 | dense : 1 | tamaño: 8 | n° filt: 16 | kernel: 10 | pool: 2 | str_pool: 2 | drop: 0.2 | epc: 100 | regL1: 0.1



Modelo n°19 | CNN 2 conv1D maxpool | V.II | Act: relu | Shift: 30 min | Lookback: 240 min | Batch: 64 | dense : 1 | tamaño: 8 | n° filt: 8 | kernel: 20 | pool: 2 | str_pool: 2 | drop: 0.5 | epc: 100 | regL1: 0.1



Arquitectura modelo n°5

Layer (type)	Output Shape	Param #
conv1d_8 (Conv1D)	(None, 231, 32)	5792
dropout_6 (Dropout)	(None, 231, 32)	0
max_pooling1d_6 (MaxPooling 1D)	(None, 231, 32)	0
conv1d_9 (Conv1D)	(None, 212, 16)	10256
dropout_7 (Dropout)	(None, 212, 16)	0
max_pooling1d_7 (MaxPooling 1D)	(None, 106, 16)	0
conv1d_10 (Conv1D)	(None, 67, 10)	6410
dropout_8 (Dropout)	(None, 67, 10)	0
max_pooling1d_8 (MaxPooling 1D)	(None, 16, 10)	0
flatten (Flatten)	(None, 160)	0
dense (Dense)	(None, 16)	2576
output (Dense)	(None, 1)	17

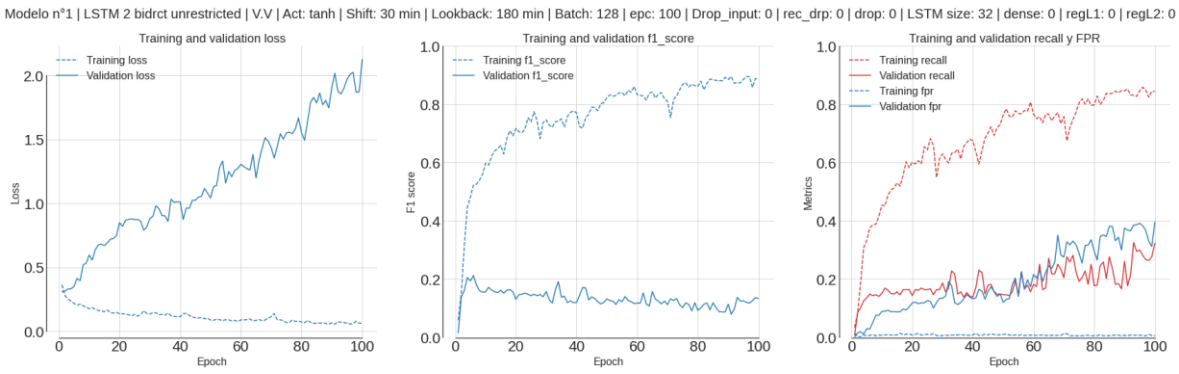
Total params: 25,051
Trainable params: 25,051
Non-trainable params: 0

Modelos LSTM versión III

Tamaño ventana: 180 min; Etiquetado: 30 min; Batch: 128

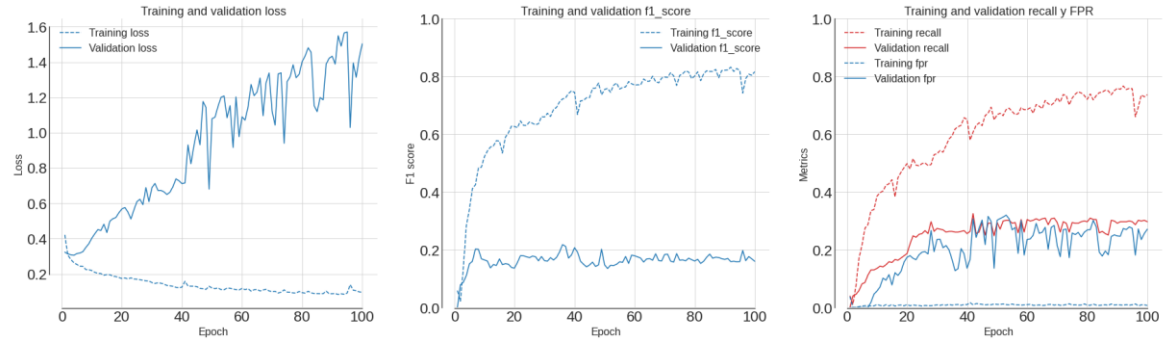
n°	Act	Dropout / Drop_Rec	Capas LSTM	Capas Densas	Estructura	Reg L1/L2	Tamaño ventana [min]	Etiquetado [min]
1	tanh	0.0/0.0	(32)(16)	NO	Restricted Stateless	0.0/0.0	180	30
2	tanh	0.0/0.0	(16)(8)	NO	Restricted Stateless	0.0/0.0	180	30
3	tanh	0.0/0.2	(16)(8)	NO	Restricted Stateless	0.0/0.0	180	30
4	tanh	0.0/0.0	(16)(8)	NO	Restricted Stateless	0.1/0.0	180	30
5	tanh	0.0/0.0	(16)(8)	NO	Restricted Stateless	0.0/0.0	180	30

Curvas de aprendizaje

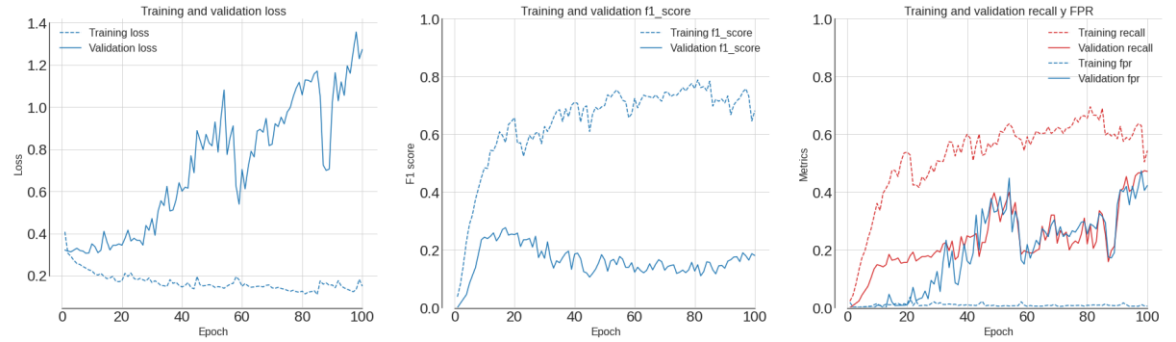


Modelo n°2 | LSTM 2 bidrct unrestricted | V.V | Act: tanh | Shift: 30 min | Lookback: 180 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

Modelo n°3 | LSTM 2 bidrct unrestricted | V.V | Act: tanh | Shift: 30 min | Lookback: 180 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0.2 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

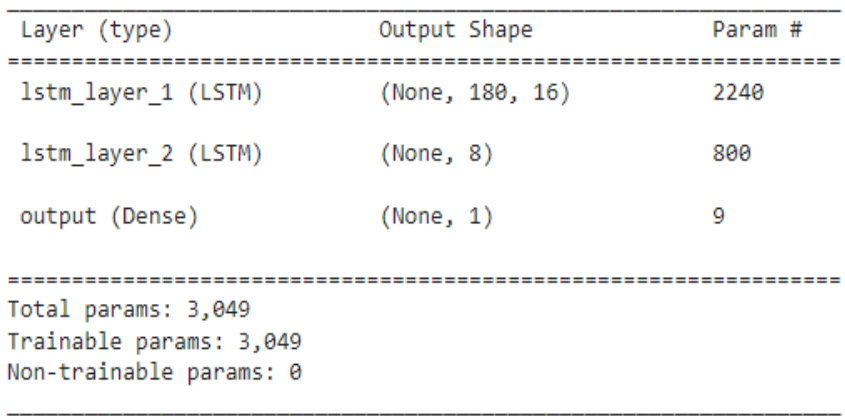


Modelo n°4 | LSTM 2 bidrct unrestricted | V.V | Act: tanh | Shift: 30 min | Lookback: 180 min | Batch: 128 | epc: 100 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0.1 | regL2: 0



Modelo n°5 | LSTM 2 RESTRICTED STATELESS BASELINE | V.V | Act: tanh | Shift: 30 min | Lookback: 360 min | Batch: 128 | epc: 50 | Drop_input: 0 | rec_drp: 0 | drop: 0 | LSTM size: 16 | dense: 0 | regL1: 0 | regL2: 0

Arquitectura mejor modelo



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_layer_1 (LSTM)         (None, 180, 16)           2240

 lstm_layer_2 (LSTM)         (None, 8)                 800

 output (Dense)              (None, 1)                 9

=================================================================
Total params: 3,049
Trainable params: 3,049
Non-trainable params: 0
_________________________________________________________________
```