Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA QUÍMICA

Mejora del control de calidad de un proceso mediante técnicas de aprendizaje automático

Autor:

Pequeño Alonso, Álvaro

Tutor:

De La Fuente Aparicio, María Jesús

Departamento de Ingeniería de Sistemas y Automática.

Valladolid, julio 2020.

INDICE DE CONTENIDOS

RESUMEN.

Con este trabajo se pretende mejorar la calidad de un proceso industrial mediante el uso de métodos de la detección y diagnóstico de fallos (FDD) o anomalías que ocurran en la planta y que alejen al producto de sus especificaciones. Debido a la digitalización de las empresas en la llamada industria 4.0, se ha dado una proliferación de sensores, lo que significa una gran cantidad de información recogida en las plantas industriales, por lo que las técnicas usadas en este trabajo están basadas en datos. El primer paso de un método FDD consiste en la detección de un fallo, y se usará una técnica de control estadístico de procesos, el análisis de componentes principales (PCA). El segundo paso, una vez detectado el fallo, es el diagnóstico, es decir, identificar el origen del fallo. Para esto se usarán técnicas de aprendizaje automático para clasificar los fallos detectados. En concreto se trabajará con redes neuronales de propagación hacia adelante con una o múltiples capas ocultas, el encadenamiento de *autoencoders* para generar una red de aprendizaje profundo (*deep-learning*) y la combinación de árboles de decisión para crear un bosque aleatorio (*random forest*). Estas técnicas se aplicarán a una planta química usada como *benchmark* en la literatura científica, la conocida Tennessee Eastman Plant (TEP), de donde se obtendrán los datos de fallo y comportamiento normal. Una vez aplicada cada técnica considerada se hace un estudio comparativo entre ellas, se extraen conclusiones y se hace un breve estudio del trabajo futuro que se puede seguir haciendo para mejorar este trabajo.

ABSTRACT

The aim of this work is to improve the quality of an industrial process by using fault detection and diagnosis (FDD) or anomalies that occur in the plant and cause the product to deviate from its specifications. Due to the digitalization of companies in the so-called industry 4.0, there has been a proliferation of sensors, which has meant a large amount of information collected in industrial plants, which is why the techniques used in this work are based on data. The first step of an FDD method consists in the detection of a fault, and a technique of statistical process control, the Principal Component Analysis (PCA), will be used. The second step, once the fault has been detected, is the diagnosis, i.e., identifying the origin of the fault. This will be done using machine learning techniques to classify the faults detected. Work will be done with feedforward neural networks with one or multiple hidden layers, the concatenation of auto-encoders to generate a deep learning network and the combination of decision trees to create a random forest. These techniques will be applied to a chemical plant used as a benchmark in the scientific literature, the well-known Tennessee Eastman Plant (TEP), from which faulty and normal behavior data will be obtained. Once each technique considered is applied, a comparative study will be made between them, and conclusions will be drawn, as well as a brief study of the future work which can be done to improve this study.

CAPITULO I:
INTRODUCCION.

1. Introducción

El control de calidad es esencial en el proceso productivo, no solo para asegurar que el producto final cumple con las especificaciones, si no que una correcta monitorización de los parámetros del proceso, y un sistema efectivo para la detección de anomalías en el mismo, puede evitar tantos daños humanos, materiales, y/o medioambientales. Para un buen control de calidad son esenciales dos acciones, una rápida detección de fallos, y un eficaz diagnóstico de estos, y de esta forma saber cómo actuar sobre la planta para devolverla a la situación de normalidad.

Los métodos de detección y diagnóstico de fallos (FDD) son usados para identificar la ocurrencia de anomalías en la planta industrial que causan al producto desviarse de las especificaciones y averiguar su localización en la planta y determinar su causa y sus consecuencias, Son necesarios en cualquier planta industrial para mantener el proceso bajo control y el producto en sus especificaciones, a la vez que se cumplen unos estándares de seguridad, tanto para los equipos como para el personal, y se garantiza la no emisión de contaminantes al medio ambiente.

En las últimas décadas, ha aparecido el concepto de industria 4.0. Esta "cuarta revolución industrial", como es clasificada por algunos autores [1], se caracteriza por una digitalización de sus sistemas, por una mayor instrumentalización del proceso, con instrumentos de medida más precisos y rápidos, y por una mejora en la interconectividad de la planta, con el uso de nuevas tecnologías de comunicación industrial. Esto ha significado una transformación del paradigma en el control de procesos, desde uno donde los conjuntos de datos eran homogéneos y univariantes, mayoritariamente escalares; hasta uno donde los datos recogidos son heterogéneos y multivariantes, representado por matrices de múltiples dimensiones [2]. Esto es el conocido "Big Data". Esta cantidad de datos es difícil de tratar e interpretar con métodos tradicionales, como el uso de modelos físicos o analíticos, por lo que surge la necesidad de aplicar nuevas herramientas en el análisis de datos, por ello de detección y diagnóstico de fallos usados en este trabajo están basados en datos.

Para la detección de fallos o anomalías en una planta, el control estadístico de procesos (SPC, por sus siglas en inglés) es el instrumento más usado a escala industrial. Proporciona las herramientas necesarias para la reducción de la variabilidad de las características principales de un producto, proporcionando un entorno para la detección de sucesos anormales. En este contexto se sitúa el análisis de componentes principales (PCA, por sus siglas en inglés), una herramienta que facilita el tratamiento de grandes cantidades de variables correlacionadas, al crear un espacio donde es clara la

visualización de esta correlación. En este análisis, se usan dos estadísticos complementarios, uno para monitorizar la variabilidad en el subespacio PCA (el estadístico Hotelling's T^2) y otro sigue la variabilidad alrededor de este subespacio (el estadístico Q), siendo función de la proyección de los residuos. Esta estructura es útil para identificar las variables responsables de los fallos y/o las variables que están más afectadas por estos. [3].

El campo con más futuro en el tratamiento de datos industriales para el diagnóstico de fallos es el relativo a la inteligencia artificial, más en concreto, las técnicas de aprendizaje automático ("machine learning"). Estas resultan efectivas en el reconocimiento de patrones y en la clasificación de datos, convirtiéndolas en ideales para el diagnóstico de las variaciones anormales en el comportamiento del proceso. Entre estas, se incluyen las redes neuronales, donde las más complejas son capaces de procesar gran cantidad de datos ("deep learning"), y los árboles de decisión, que se pueden combinar para formar bosques aleatorios y así aumentar la capacidad de tratamiento de datos.

2. Objetivos

En este trabajo se plantea desarrollar métodos de detección y diagnosis de fallos (FDD, por sus siglas en inglés), metodologías aplicables en todo tipo de industrias, ya que en todas se pueden producir fallos que afectan a la fiabilidad del proceso, la seguridad de la planta, y a la calidad de los productos. Además, junto a la aparición de conceptos como industria 4.0, aparecen nuevas necesidades de procesamiento de gran cantidad de datos, idea en la que están centrados los métodos de FDD desarrollados en este proyecto.

El primer paso de un método FDD es la detección del fallo o anomalía en la planta. Para resolver este problema se utilizarán técnicas de control estadístico de procesos, y en concreto el análisis de componentes principales. Una vez detectado, es necesario diagnosticar que fallo ha ocurrido, o lo que es lo mismo, identificar el elemento o elementos de la planta que presentan problemas en su funcionamiento. Para este segundo paso se usarán también técnicas basadas en datos, y en concreto, técnicas de inteligencia computacional (redes neuronales , *deep-learning* y bosques aleatorios) que se usarán como técnicas de clasificación. Se hará una comparativa entre las diferentes técnicas de diagnosis, para sacar las conclusiones correspondientes.

Los datos sobre los que se trabajará provienen de la planta Tennessee Eastman, un *benchmark* sacado de la literatura científica que sirve de referencia en el estudio del control y monitorización de procesos multivariantes.

3. Organización

La memoria de este trabajo está organizada en seis capítulos.

En el primer capítulo, se realizará una introducción al tema tratado, proporcionando el contexto y justificaciones necesarias para englobar este trabajo en el marco de los métodos de detección y diagnóstico de fallos

En el segundo capítulo se realiza un estudio teórico sobre el control de calidad, la detección de fallos mediante el control estadístico de procesos, y el diagnostico de fallos mediante técnicas de aprendizaje automático.

En el tercer capítulo se introduce la planta Tennessee Eastman y se describe su historia, funcionamiento y variables. También se presentan los fallos que se utilizarán durante la experimentación

En la cuarta capítulo se expone la experimentación desarrollada para este trabajo, explicando los pasos seguidos para desarrollar el análisis de componentes principales para la detección de fallos, y la creación de las técnicas de aprendizaje automático para su diagnóstico. También se presentan los resultados de cada caso de estudio y se realiza un estudio comparativo entre ellos.

Por último, se resumen las conclusiones de este trabajo y los resultados derivados de la experimentación, acabando con sugerencias para futuros estudios sobre este tema.

CAPITULO II: ESTUDIO TEORICO

1. Control de calidad

El control de calidad es todo método orientado a ayudar en la toma de decisiones relacionadas con el cumplimiento de los requerimientos de producto y seguridad. Los requerimientos mínimos que habrá que cumplir son las especificaciones del producto, y la garantía de un proceso seguro según los estándares, pero no solo hay que cumplir lo mínimo; es de interés que el producto y la planta estén en condiciones satisfactorias, asegurarse de que sean constantes y poco variables, y en especial que sean económicamente viables. Es un aspecto esencial en la manufactura, desde el comienzo del tratado de las materias primas, hasta la salida del producto final de la planta. Especialmente, es importante en la ingeniería química, donde los productos y procesos son más sensibles a los errores en los mismos, y donde frecuentemente se trabaja con compuestos y condiciones de operación peligrosas, tanto para el personal, como para el medioambiente. Asegurar un buen control de calidad es asegurar la minimización de accidentes y de vertidos indeseados al entorno, así como garantizar que el producto esté dentro de las especificaciones requeridas.

En el control de calidad se puede englobar, en orden, la monitorización del proceso, en la cual se detectan los fallos, el análisis de los fallos, donde se diagnostican, y la actuación sobre el proceso para devolverlo al estado de normalidad. Los requisitos fundamentales para un buen control de calidad son: instrumentos de medida precisos, un sistema de comunicación industrial que soporte la toma de datos, una temprana detección de los fallos y un diagnóstico correcto, para identificar el origen del fallo, y las variables causantes y afectadas [4]. Conviene entender la definición de fallo en este contexto. Un fallo en un entorno industrial está definido como una condición anormal o defecto en un componente, equipamiento o subsistema que conlleva una situación de no cumplimiento de las especificaciones requeridas, riesgo, o daño material o personal (ISO 10303-226).

Para monitorizar el proceso se requieren mediciones en diferentes partes de las etapas productivas para evaluar su funcionamiento. Esto puede ser mediante sensores que midan variables relacionadas con la etapa correspondiente, o mediante toma de muestras aleatorias del producto en esa fase. La calidad está ligada al porcentaje de producto que queda fuera de los límites de especificación, y pueden estudiarse mediante distribuciones de probabilidad. La variabilidad en el proceso es la razón por la que las especificaciones no se cumplan, siendo las causas más comunes la inherente falta de replicabilidad existente en todo proceso industrial, debido a la imposibilidad de controlar la gran cantidad de variables que influyen en él, y las perturbaciones ajenas al proceso.

La detección y diagnóstico de fallos (FDD) es el proceso de descubrir la existencia de un fallo (detección) y su posterior identificación del origen, causas (diagnóstico) para determinar las consecuencias que ha tenido (evaluación). Se puede dividir en tres grandes partes, la detección del fallo, que determina si se produce o no, el diagnostico, que a su vez está dividido en dos fases, el aislamiento del fallo y su identificación, que incluye determinar el alcance del fallo y su comportamiento, y por último la evaluación de daños que haya podido causar el fallo. El control estadístico de procesos (SPC) es el método más usado en el control de calidad, detectando fallos en procesos industriales, ya que proporciona herramientas útiles y eficaces para tratar la variabilidad del proceso. Por otro lado, las técnicas de aprendizaje automático están cobrando especial importancia en el ámbito del diagnóstico de fallos debido a su potencial para tratar las grandes cantidades de datos generadas en una planta industrial moderna, y clasificar estos datos de comportamiento normal de la planta o de comportamiento defectuoso.

2. Detección de fallos mediante control estadístico de procesos.

Se entiende como detección de fallos al aviso de un funcionamiento anómalo en los componentes del sistema, sean sensores, actuadores o equipos. [5]. Conocer la existencia de una anomalía en el proceso no implica que se entiendan sus causas o consecuencias. La detección únicamente implica la determinación de la existencia de la anomalía. Para esto, es muy útil las herramientas desarrolladas en el contexto del control estadístico de procesos (SPC).

El SPC es un método de control de calidad basado en herramientas estadísticas para monitorizar y controlar un proceso. El SPC se puede aplicar a cualquier proceso donde el producto tenga unas especificaciones medibles. Las ventajas de este método respecto a otros tipos de control de calidad, como la inspección del producto, es la detección temprana y la prevención de problemas, en contraste con la corrección de estos una vez ocurridos. Además de maximizar el aprovechamiento de los recursos, un SPC correctamente implementado puede reducir el tiempo y el coste en la producción.

2.1. Variabilidad del proceso

En 1924, el Dr. Shewart se percató de que los datos provenientes de las mediciones de las variaciones en el proceso productivo pocas veces producen

una curva de distribución normal o campana de Gauss, no como los fenómenos naturales, que tienden a organizarse de esta forma. Todos los procesos tienen variaciones, pero mientras unos muestran solo variaciones naturales del proceso (causas comunes de variación), estando bajo control estadístico, otros presentan variaciones a mayores de las inherentes del proceso (causas especiales de variación), y se denominan fuera de control [6]. Cualquier variación en un proceso se podrá clasificar en estos dos tipos. Las causas comunes de variación son fuentes normales de variación, actuando de forma consistente en el proceso y al acumularse dan lugar a una distribución estadística estable y replicable a lo largo del tiempo. Las causas especiales de variación pueden ser asignadas a una perturbación concreta del proceso, y por naturaleza son impredecibles e intermitentes. Estas causas no controlables en el proceso son las que producirán los fallos que habrá que detectar y diagnosticar.

Para determinar si el producto o proceso están dentro de las especificaciones hace falta definir los limites superior e inferior, que establecen una zona dentro de la cual se cumplen las especificaciones, y fuera de ella no. Se distinguen entre limites naturales del proceso (LPL, límite inferior del proceso y UPL, límite superior del proceso), donde este es estadísticamente normal, y los límites de especificaciones (LSL, límite inferior de especificaciones, y USL, límite superior de especificaciones). Cuando los límites de las especificaciones son mayores que ambos límites naturales del proceso, se dice que el proceso es tanto estable como capaz, y producirá con alta fiabilidad el producto correcto, por lo que se podrán hacer predicciones acertadas sobre el comportamiento del proceso. Cuando el proceso es estable pero los límites de control son mayores que uno o ambos límites de especificación, el proceso es estable pero no es capaz y aparecen errores en la producción, y el sistema es incapaz de producir con la calidad adecuada, como se refleja en la figura 1. Por el contrario, cuando se presentan causas de variación especiales (fallos), la salida del proceso no es estable a lo largo del tiempo y tampoco es predecible, porque no se sabe cómo se comportará el proceso. En estos casos se dice que el sistema está fuera de control estadístico como se muestra en la figura 2.

Los datos de las mediciones se comparan con los límites superior e inferior de tolerancia establecidos y si se encuentran entre estos límites el proceso funciona correctamente. Si están fuera, el proceso necesita ser ajustado para devolver al producto a las especificaciones. Para realizar este ajuste es importante distinguir entre los conceptos de precisión y exactitud. Precisión se refiere a la dispersión del conjunto de valores obtenidos de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Está asociada a la desviación estándar de las mediciones. En la

actualidad, debido a la automatización de los procesos, estos tienen una precisión alta. Exactitud es la cercanía del valor real al valor medido. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. La exactitud se expresa mediante el error absoluto, que es la diferencia entre el valor experimental y el valor verdadero. Para determinar estas dos características hay que analizar un conjunto de datos obtenidos a lo largo del tiempo, no vale con estudiar observaciones individuales.

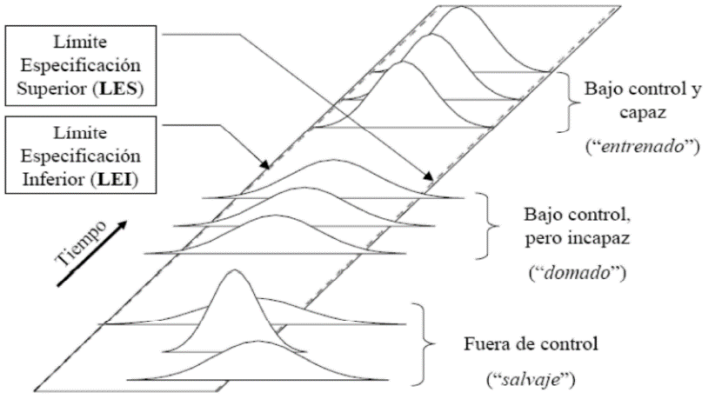

Figura 1.- De arriba abajo, un proceso capaz y controlado, un proceso incapaz pero controlado, y un proceso fuera de control

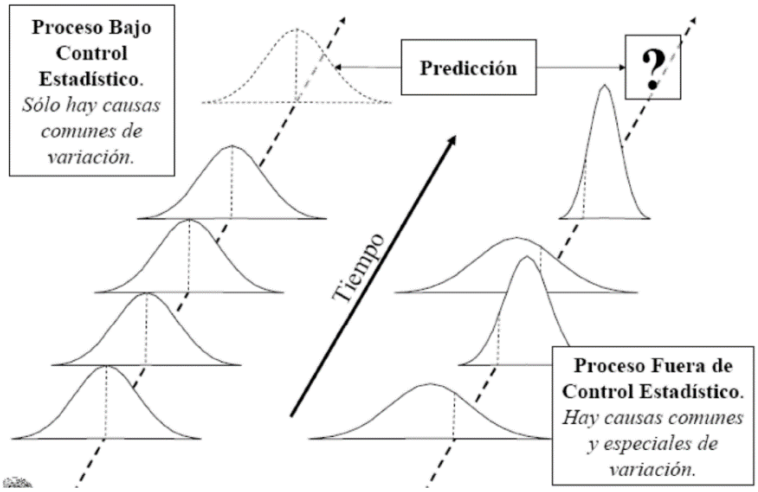

Figura 2.- A la izquierda, un proceso bajo control fácilmente predecible. A la derecha, las causas especiales de variación impiden las predicciones [3]

2.2. Gráficos de control

Una de las herramientas para analizar la variabilidad de un proceso son los gráficos de control. Son diagramas que muestran los valores producto de la medición de una característica de calidad, ubicados en una serie cronológica.

2.2.1. Gráficos Shewhart

El Dr. Shewhart, una vez definidas las distinciones entre causas comunes y especiales de variación. desarrolló un simple método para detectar la presencia de los factores asignables que desviaban la producción del estándar a alcanzar. Este método fue el de los gráficos de control, que monitorizan la evolución temporal de algún parámetro e indican visualmente cuando sobrepasa alguno de los límites establecidos. Si el análisis del gráfico de control indica que el proceso está actualmente bajo control, es decir, es estable, con variaciones que sólo provienen de fuentes comunes al proceso, entonces no se necesitan correcciones o cambios en los parámetros del proceso. Además, los datos del proceso pueden utilizarse para predecir el futuro rendimiento de este. Si el gráfico indica que el proceso no está bajo control estadístico, el análisis del gráfico puede ayudar a determinar las fuentes de variación. Un proceso que es estable, pero que funciona fuera de los límites deseados debe mejorarse mediante un esfuerzo deliberado por comprender las causas del rendimiento actual y mejorar el proceso.

Un gráfico de control (figura 3) consta de puntos que representan una variable estadística (media, un rango, una proporción) de las mediciones de una característica de las muestras tomadas del proceso en diferentes momentos. el gráfico debe tener:

- una línea central en el valor de la media de la estadística
- los límites de control superiores e inferiores indicando el umbral en el que el resultado del proceso se considera estadísticamente improbable, y se dibujan típicamente a una distancia de tres desviaciones estándar de la línea central.

El gráfico puede tener otros elementos opcionales, como:

- los límites superiores e inferiores de advertencia o control, dibujados como líneas separadas, típicamente a dos desviaciones estándar por encima y por debajo de la línea central, usados para alertar al operario de que el proceso está empezando a alejarse de la normalidad.

- la división en zonas, con la adición de reglas que rigen las frecuencias de las observaciones en cada zona, y anotaciones con eventos de interés, según lo determinado por el Ingeniero de Calidad a cargo de la calidad del proceso.

En el ámbito de los gráficos de control es común desarrollar un conjunto de normas sobre las medidas a tomar frente a las causas especiales. Por ejemplo, si se detectan siete puntos consecutivos por encima o por debajo de la línea central, se debe parar la producción, comprobar la totalidad de esta, y ajustar el proceso, tras lo cual se comprueban cinco muestras consecutivas y si las medidas están en el umbral de control estadístico del proceso, se continua con el proceso.

Por lo tanto, la finalidad de los gráficos de control es permitir una detección sencilla de los acontecimientos que son indicativos de un cambio en el proceso. La finalidad de añadir límites de alerta o subdividir el gráfico de control en zonas es proporcionar una notificación temprana si algo no funciona como es debido. En estas ocasiones de alarma temprana. en lugar de iniciar inmediatamente medidas de mejora del proceso para paliar las causas especiales, el ingeniero de calidad puede aumentar temporalmente el ritmo de toma de muestras de la salida del proceso hasta que esté claro que el proceso está realmente fuera de control [7].

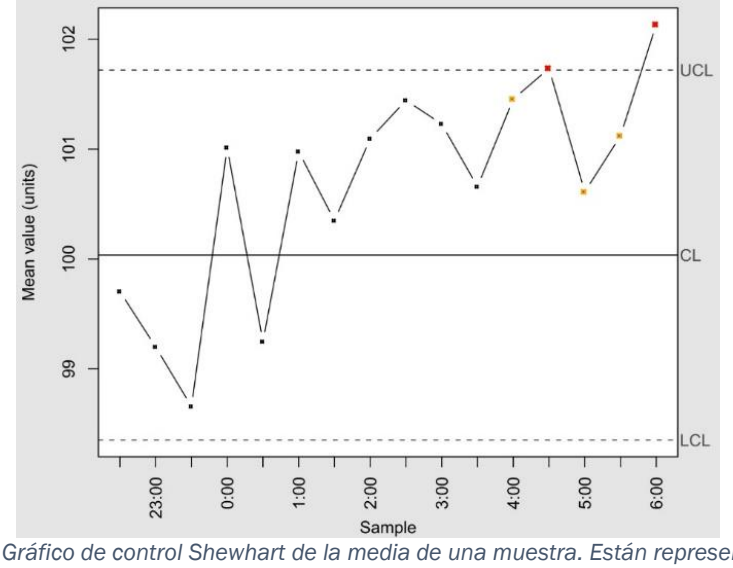

Figura 3.- Gráfico de control Shewhart de la media de una muestra. Están representados los límites de control superior (UCL) e inferior (LCL) así como la línea central indicando la media de las medias de la muestra (CL) [8].

La elección de colocar los límites a una distancia de tres desviaciones estándar no es casual. El Dr. Shewhart se basó en tres normas estadísticas:

- el resultado aproximado de la desigualdad de Chebyshev, que indica que, para cualquier distribución de probabilidad, la probabilidad de un resultado superior a k desviaciones estándar de la media es como máximo $\frac{1}{k^2}$;
- el resultado de la desigualdad de Vysochanskii-Petunin, según el cual, en cualquier distribución de probabilidad unimodal, la probabilidad de un resultado mayor que k desviaciones estándar de la media es como máximo $\frac{4}{9k^2}$;
- en la distribución Normal, donde, el 99,7% de las observaciones se producen dentro de tres desviaciones típicas de la media. [6]

Aun basándose en la estadística para crear los límites de especificación, Shewhart argüía que la mejor forma de establecerlo en cada proceso era la práctica, ya que, como ya se ha mencionado, no tardó en percatarse de que los procesos industriales rara vez siguen una ley de distribución normal como los procesos en las ciencias naturales. Por lo tanto, los gráficos de control se concibieron como un método heurístico, esto es, basado en la experiencia.

Los gráficos de control Shewhart tienen limitaciones. Al estar situados los límites de forma que contengan el 99.7% de las observaciones, hay aproximadamente un 0,3% de probabilidad de que un punto exceda los límites de control, aunque no sea debido a una causa especial. Así pues, incluso un proceso bajo control estadístico trazado en un gráfico de control diseñado correctamente señalará eventualmente la posible presencia de una causa especial, aunque no se haya producido en la realidad. Para un gráfico de control Shewhart que utiliza límites alejado tres desviaciones estándar, esta falsa alarma se produce en promedio una vez cada 1/0,0027 (370,4) observaciones [7]. Además de falsos positivos, si se produce una causa especial, puede no ser de suficiente magnitud para que el gráfico produzca una condición de alarma inmediata. Los gráficos de Shewhart son bastante buenos para detectar grandes cambios en la media o la varianza del proceso, Sin embargo, para cambios más pequeños, el gráfico Shewhart no los detecta de manera eficiente, habiendo un retardo en la respuesta. Se han desarrollado otros tipos de gráficos de control, como el gráfico CUSUM, que detectan los cambios más pequeños de manera más eficiente utilizando la información de las observaciones recogidas antes del punto de datos más reciente.

2.2.2. Gráficos CUSUM

Al igual que los gráficos Shewhart, esta herramienta permite la detección simple y directa de causas especiales en el proceso que den origen a fallos. Los gráficos CUSUM son una reinterpretación de los gráficos anteriores

orientados a detectar pequeñas variaciones en las variables medidas. Se pueden usar de forma complementaria a otros tipos de gráficos.

En el gráfico se representa la suma acumulada de todas las desviaciones respecto a la media de la estadística graficada. Además, cuenta con los siguientes elementos:

- las observaciones normalizadas, Z_i, centradas alrededor de la media ($\bar{x}$) y baremadas con la desviación estándar
$$Z_i = \frac{\bar{X}_\iota - \bar{x}}{\sigma_i}$$
- las siguientes sumas acumuladas, S_H, que detecta una anomalía positiva, y S_L, que detecta una anomalía negativa:
$$S_{H_{i+1}} = \max(0, S_{H_i} + Z_i - k)$$
$$S_{L_{i+1}} = \max(0, S_{L_i} - Z_i - k)$$
- un valor de referencia, k, definido de forma que detecte una variación de la mitad de la desviación estándar.
- un umbral, h, para detectar fallo y dar la alarma, definido según S_H, y S_L

Las ventajas de este tipo de gráficos son la rapidez para detectar pequeñas variaciones de las variables medidas y permiten una identificación visual de cambios en la media del proceso. Las desventajas son la lentitud para detectar cambios grandes en el proceso, y que, debido a la naturaleza acumulada de los datos representados, existe una correlación entre estos y es difícil interpretar patrones.

2.3. Análisis de Componentes Principales (PCA)

Hasta ahora, el control de procesos ha sido univariante, por lo que solo se puede analizar una variable por cada gráfico, además de no estar teniéndose en cuenta la correlación entre variables, siempre presente en el contexto industrial. Esto limita bastante la utilidad de estos gráficos a conjuntos de datos pequeños y simples. El control de procesos estadístico multivariante se desarrolla para monitorizar el conjunto completo de las variables del proceso. En este contexto se sitúa el PCA.

El PCA es una técnica de proyección que usa transformaciones ortogonales para convertir un conjunto de variables correlacionadas en un conjunto de valores linealmente no correlacionados llamados componentes principales. Produce una reducción de la dimensionalidad del conjunto de datos originales, preservando su estructura de correlación. Básicamente, se

ajusta un elipsoide de *a* dimensiones al conjunto de datos con dimensiones *m* (*m*>*a*), donde cada eje del elipsoide representa un componente principal (figura 4). Los ejes de menor tamaño representan una baja varianza a lo largo de ese eje, por lo que si se omite ese eje con su correspondiente componente principal de la representación de los datos, solo se pierde una cantidad igual de pequeña de información [9]. De modo ideal, se buscan *a* < *m* variables que sean combinaciones lineales de las *m* originales y que no estén correlacionadas, recogiendo la mayor parte de la variabilidad de los datos.

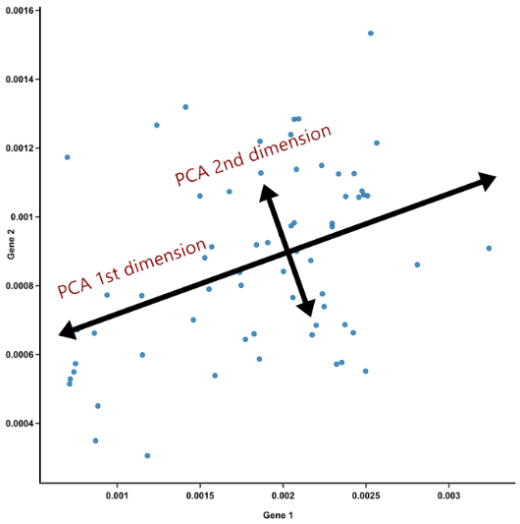

Figura 4.- Ejes de un elipsoide de dos dimensiones ajustado a un conjunto de datos

Antes de comenzar el análisis, se necesita un conjunto de datos representativo del comportamiento normal. Hay que realizar un pretratamiento de estos datos, eliminando las variables que no sean representativas, como las que se originan de una mala medición. También se deben normalizar las variables para que tengan el mismo peso en la monitorización que en el análisis. Para ello, se resta a cada variable su media y se divide entre su desviación estándar, para que la varianza sea uno.

Partiendo de los datos a analizar, se tienen *m* variables y *n* observaciones de cada variable, se construye la matriz *X* de dimensiones *n*x*m*, tal que $X \in \mathrm{R}^{nxm}$. A partir de esta matriz se calculan la matriz de correlación de X, A:

$$A = \frac{1}{(n-1)} X^T \cdot X \qquad (1)$$

A partir de esta matriz A se extraen los valores singulares, obteniendo la matriz Λ, una matriz diagonal que contiene los valores propios reales no negativos de A en orden decreciente en su diagonal principal. Con A también

se consigue los vectores propios, llamados vectores de carga, que se almacenan en una matriz V, en sus columnas

$$A = V\Lambda V^T \tag{2}$$

Con el objetivo de conseguir la máxima representación de las variaciones de los datos sin que pesen los efectos de ruido, se retienen un número, *a*, de vectores de carga, correspondientes a los *a* valores singulares más grandes. Este número será menor al número de variables, *m.* Se guardan estos vectores en una matriz P de dimensiones *mxa*. Con esto se proyecta el conjunto de datos de observación $X \in R^{nxm}$ en $T \in R^{nxa}$:

$$T = X \cdot P \tag{3}$$

Las *a* variables transformadas se llaman componentes principales y las observaciones individuales, *x*, transformadas son los *scores.*

Acabada la proyección, se puede calcular la matriz de residuos, recalculando los datos originales en función de T

$$\hat{X} = T \cdot P^T \tag{4}$$

y definiendo una matriz de residuos, calculada como la diferencia entre la matriz original, X, y la recalculada, $\hat{X}$:

$$E = X - \hat{X} \tag{5}$$

La matriz de residuos, E**,** captura la variación de los datos de observación contenidos en los vectores carga asociados con los m-a valores singulares más pequeños, de tal forma que se puede definir la matriz original X como:

$$X = T \cdot P^T + E \tag{6}$$

Para escoger el número de vectores de carga retenidos, que será el tamaño del nuevo espacio de proyección hay varios criterios:

- test del porcentaje de varianza, con el que se selecciona *a* de forma que explique un porcentaje específico de la varianza total
- test del codo, en el que se representan los valores de los autovalores frente a las observaciones que les corresponden y se busca un punto de inflexión en la gráfica, a partir del cual todos los valores de los autovalores son parecidos y pequeños.

Construyendo este método se cumplen las siguientes propiedades. Definiendo t_i como la columna i-ésima de T:

- La varianza está ordenada de mayor a menor: var(t_1) > var(t_2) > … > var(t_a)
- Está centrada en la media: media(t_i) = 0;
- Descomposición ortogonal: $t_i^T t_j = 0, \forall\ i \neq j$
- No existe ninguna otra expansión ortogonal de *a* componentes que capture más variación de los datos.

Una vez creado el modelo, para detectar los datos se deben definir los umbrales T^2 y Q a partir de sus respectivos estadísticos. La estadística T^2 o Hotelling's se utiliza en el espacio de dimensión a, para detectar comportamientos anómalos del sistema cuando traspasan un umbral. La estadística Q se usa para monitorizar el resto del espacio de observación correspondiente a los *m-a* valores singulares más pequeños, es decir para monitorizar el espacio de los residuos.

Para calcular la estadística T^2 a partir de *a* componentes principales se utiliza la ecuación:

$$T^2 = x \cdot P \cdot \Lambda^{-1} \cdot P^T \cdot x \tag{7}$$

Cuando se calcula para una observación *x* de *m* variables, puede interpretarse como la distancia de la observación al centro del modelo. Los *scores* están escalados inversamente proporcional a la varianza. Esto permite definir un umbral T_α^2 escalar característico de la variabilidad en todo el espacio a-dimensional. Cuando se supere este umbral, se habrá detectado un fallo o habrá habido una falsa alarma:

$$T_\alpha^2 = \frac{(n^2-1)}{n \cdot (n-a)} \cdot F_\alpha(a, n-a) \tag{8}$$

donde F_α es la distribución de Fisher-Snedecor, con *a* y *n-a* grados de libertad y donde α es el grado de fiabilidad, que en el contexto de la detección de fallos es el porcentaje de falsas alarmas y fallos no detectados [10].

Como T^2 está enfocado a los valores singulares más grandes, se puede afinar la detección teniendo en cuenta el error de predicción o estadístico Q, que se calcula a partir de los residuos como:

$$Q = r \cdot r^T \tag{9}$$

donde r es el vector de residuos:

$$r = x - \hat{x} = x \cdot (I - P \cdot P^T) \tag{10}$$

El umbral se calcula aplicando la distribución Chi-cuadrado con una probabilidad del 99% y $2 \cdot \frac{\mu_Q^2}{\sigma_Q^2}$ grados de libertad, siendo μ_Q la media estadística del estadístico Q, y σ_Q su desviación estándar :

$$Q_\alpha = \frac{\sigma_Q^2}{2 \cdot \mu_Q} \cdot Chi\left(p, 2 \cdot \frac{\mu_Q^2}{\sigma_Q^2}\right) \tag{11}$$

La detección de los fallos se realiza con mayor exactitud si se utilizan ambos estadísticos, debido a que T^2 detecta cambios alejados del origen del sistema, mientras que Q captura el ruido, por lo que para determinados fallos puede ser un estadístico más adecuado que otro [10].

Con los vectores de residuos calculados (ecuación 10) se puede obtener un diagrama de contribuciones que indica que variables del espacio original son las más influyentes en el fallo detectado (figura 5).

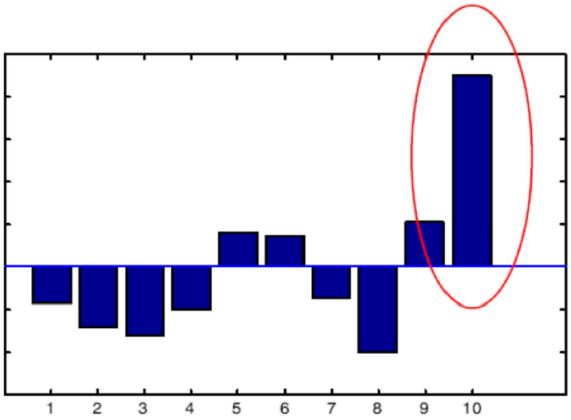

Figura 5.- Diagrama de contribuciones. La variable 10 es la mayor responsable del fallo detectado [3].

3. Diagnóstico de fallos mediante técnicas de aprendizaje automático.

El diagnóstico de los fallos se realiza después de haberse detectado una anomalía en el proceso, para caracterizar que tipo de fallo ha habido, es decir, diagnosticar la causa u origen del fallo. En el contexto de las técnicas de aprendizaje automático se entiendo como diagnóstico a la clasificación de los fallos en diferentes clases (fallo 1, fallo 2, fallo 3, etc.) mediante el aprendizaje de patrones y la posterior clasificación de forma automática,

3.1. Aprendizaje automático.

El aprendizaje automático o *machine learning* es una disciplina en el ámbito de las ciencias computacionales, y una rama de la Inteligencia Artificial, que estudia posibles técnicas para desarrollar algoritmos que se mejoren a sí mismos automáticamente. Estos algoritmos de aprendizaje desarrollan un modelo matemático basado en un conjunto de datos de muestra, o datos de entrenamiento. El objetivo último es que los modelos creados sean capaces de hacer predicciones o tomar decisiones a partir de datos reales, sin haber sido creados específicamente para ello [11].

Existen dos tipos fundamentales de aprendizaje automático, no supervisado y supervisado. En el no supervisado, no se proporciona una salida deseada, y se deja al algoritmo que encuentre una regla general que defina a los datos. En el supervisado, se suministra al algoritmo tanto datos de entrada como los resultados deseados, para que defina un modelo que transforme las entradas en salidas. Además, dentro del aprendizaje supervisado, se pueden realizar modelos de clasificación, que asignan categorías a los conjuntos de datos, o de regresión, que predicen un número basándose en las tendencias de los datos.

Un problema típico en el aprendizaje automático es el sobreajuste. Este concepto tiene que ver con crear un modelo demasiado ajustado a un conjunto concreto de datos, de tal forma que con nuevos datos tenga problemas. El error principal consiste en crear un modelo demasiado complejo intentando explicar idiosincrasias que no aportan valor a las capacidades predictivas del modelo. Por ejemplo, en la figura 6, un modelo sobre ajustado corresponde a la línea negra, que, aunque sigue perfectamente a los datos representados, si se introducen nuevos datos la línea del modelo es probable que no vaya a representarlos correctamente. Por eso, la línea azul sería un modelo más correcto, porque, aunque no representa a la perfección el modelo, tiene más flexibilidad para hacer predicciones con nuevos datos.

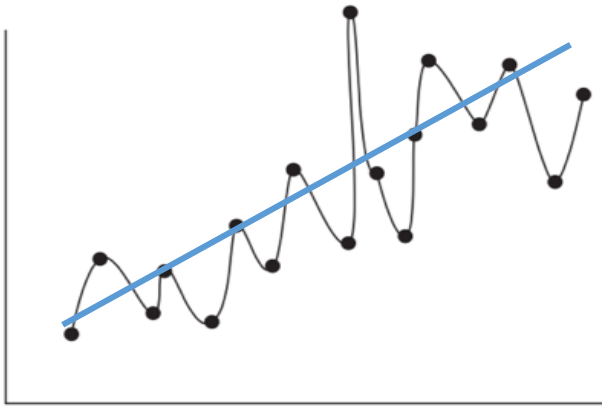

Figura 6.- Sobreajuste de un modelo

Para evitar el sobreajuste, lo normal es dividir los datos de entrada y hacer pasar la técnica de aprendizaje automático por tres etapas: entrenamiento, prueba y validación. A continuación, se explican estos tres conceptos.

En el entrenamiento, se emplean la mayor parte de los datos, con la finalidad de proporcionar al modelo datos de referencia con los que pueda aprender a reconocer tendencias o patrones. Para optimizar el modelo se tiene que minimizar la función de pérdida, que representa el error del modelo al procesar los datos.

Después de haber entrenado y haberse creado un modelo, se prueba con una pequeña parte de los datos que aún no han sido vistos por la técnica de aprendizaje. Normalmente en esta etapa se realizan ajustes más precisos a los parámetros del modelo. Comparando la función de pérdida de la prueba frente a la del entrenamiento se pueden detectar sobreajustes, si una disminuye, pero la otra no se ve afectada. Si esto ocurre, se debe modificar el modelo y volver a entrenarlo.

Por último, la validación se realiza con los datos restantes, para asegurarse de minimizar las coincidencias de que el modelo funcione con los datos de entrenamiento y prueba. Idealmente, esto se debe realizar una única vez para obtener una impresión de la calidad del modelo.

En el ámbito de las técnicas de aprendizaje automático de clasificación, las cuales se estudian en este trabajo, existen los conceptos de atributo y clase. Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables, y cada objeto particular puede tener valores distintos para estas variables. Las clases son las categorías en las que el modelo divide a los datos para clasificarlos.

3.2. Redes neuronales artificiales.

Las redes neuronales artificiales son modelos computacionales basados en las redes neuronales del cerebro. Son entramados de nodos llamados neuronas, que reciben señales, las procesan, y pueden enviar la señal procesada hacia otras neuronas. Estas neuronas se agrupan en capas interconectadas: la capa que recibe los datos de entrada es la capa de entrada, y la capa donde se produce el resultado final es la capa de salida. Entre ellas, puede no haber nada o haber una o varias capas ocultas, que toman entradas y las aplican transformaciones para producir salidas. Las neuronas de las capas ocultas pueden estar conectadas de varias formas con las neuronas de capas ocultas anteriores y posteriores, siendo la más común la total conexión

de las neuronas, en las que una neurona de una capa envía su salida a todas las neuronas de la siguiente capa.

Hay dos parámetros fundamentales en las redes neuronales, el peso (w) y el *bias* (b). El peso está asignado a cada una de las conexiones entre nodos, y multiplica al valor de entrada. Es una forma de ponderar las entradas, y representa la influencia que tendrán los cambios en estas: un peso alto modificará de manera significativa la salida. El *bias* es un parámetro con el que cuenta cada neurona, y que se suma al valor de entrada. Representa como de lejos está el resultado real del deseado: un *bias* bajo indica que el valor de la salida es similar al del valor deseado. Ambos parámetros son modificados por la propia red durante su entrenamiento.

Por lo tanto, el estado activado de una neurona, llamado nivel (*n*), se obtiene de multiplicar a la entrada, x, por el peso, y sumándole, si corresponde, un *bias* constante:

$$n = w \cdot x + b$$

La neurona lleva asociada una función de transferencia o de activación, F, que se aplica al nivel calculado, para obtener la salida, y, que se transmitirá a la siguiente salida. La función de transferencia puede ser de cualquier tipo, siendo las más usadas las tipo escalón, lineales, sigmoides o gaussianas [12].

$$y = F(n) = F(w \cdot x + b)$$

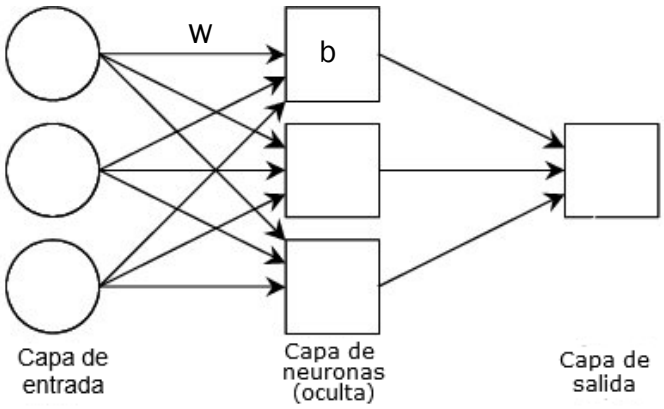

Figura 7.- Esquema de una red neuronal simple

3.2.1.Red de propagación hacia delante. Perceptrón y multicapa.

Una red de propagación hacia delante es una red neuronal en la que las conexiones entre los nodos no tienen recirculación alguna. Son el tipo más

simple de red neuronal, y pueden dividirse en redes de propagación hacia delante sin capa oculta o *perceptrón* y redes con una o más capas ocultas.

Un *perceptrón* se considera el modelo de red neuronal artificial más simple, con una capa de entrada y una de salida. Es un algoritmo de aprendizaje supervisado de clasificación binaria (agrupa los elementos de un conjunto en dos clases diferentes según una regla de clasificación). Se desarrolló en 1958 por Frank Rosenblatt, con el fin de usarse para el reconocimiento de imágenes. Aunque prometedor, pronto se puso en evidencia que solo era capaz de reconocer patrones sencillos. Esto es debido a que son clasificadores lineales, esto es, usan combinaciones lineales de las entradas para la toma de decisiones, lo que limita considerablemente al algoritmo. A pesar de sus limitaciones, ha sido demostrado que algunos modelos lineales producidos por el *perceptrón* se asemejan bastante a un neurona real [13].

El funcionamiento del *perceptrón* es simple. Por ejemplo, tómese un perceptrón con cinco entradas sin *bias* (figura 8) [14]:

1. Cada una de las señales de entrada (x_i) es multiplicada por su *peso* (w_i), y, si hubiera, se le sumaría su *bias.*
2. Usando una función sumadora, todos los valores resultantes del paso anterior se suman:

$$\sum_{1}^{5} x_i \cdot w_i \,(+b_i)$$

3. Se aplica a la suma una función de activación para adecuar el resultado al rango de números con el que se esté trabajando. Lo más común es usar una función escalón (1 ó 0), ya que los *perceptrones* son clasificadores binarios.

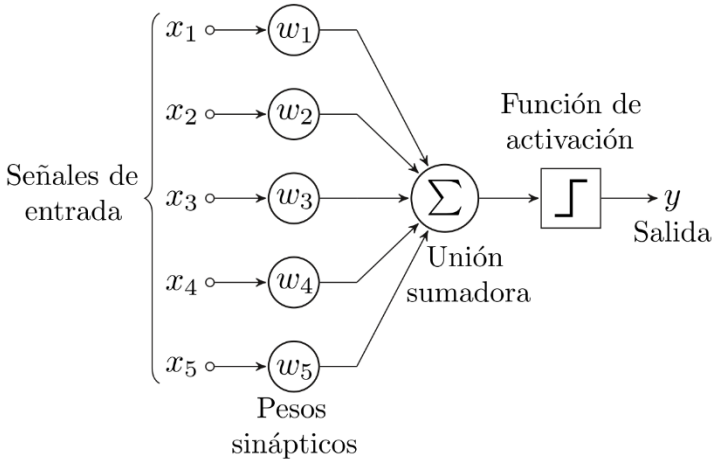

Figura 8.- Diagrama de un perceptrón con cinco señales de entrada [14].

Una red de propagación hacia delante multicapa posee una capa de entrada, una o más capas ocultas de neuronas y una capa de salida. Exceptuando a los nodos de entrada, el resto son neuronas que usan funciones de activación no lineales. Son más complejas que los *perceptron*, usando algoritmos de entrenamiento basados en el cálculo de gradientes. Este método de entrenamiento se denomina retropropagación, y consiste en la comparación de la señal de salida deseada con cada una de las salidas reales. El error se propaga hacia las capas inmediatamente anteriores, y se reparte entre cada neurona basado en la contribución de estas en la salida original. A medida que se entrena la red, se modifican los parámetros (pesos y *bias*) para minimizar el error.

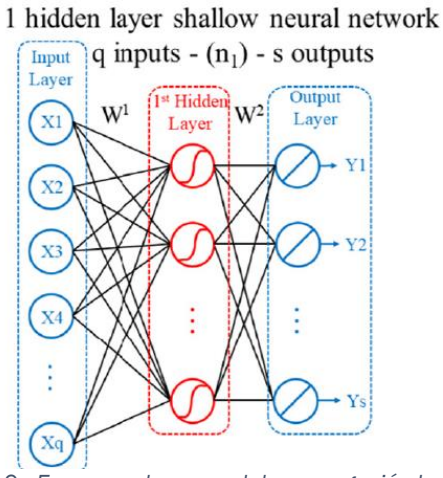

Figura 9.- Esquema de una red de propagación hacia delante con una capa oculta [15].

Entre las ventajas de las redes multicapa, destaca que pueden usarse para aproximar cualquier función, como se demuestra con el teorema de Cybenko [15], por lo que pueden usarse para la creación de modelos de análisis de regresión. Como la clasificación es un tipo de regresión en la que las salidas son categóricas, las redes de propagación hacia delante multicapa son buenos algoritmos de clasificación. Esto las convierte en ideales para la detección de fallos.

3.2.2.Autoencoders. Deep learning.

Los *autoencoders* son un tipo especial de red de propagación hacia delante multicapa, donde la entrada es la misma que la salida. No se programan para que copien la entrada perfectamente, con esto solo se conseguiría duplicar la señal. Los *autoencoders* comprimen la entrada a un espacio de menor dimensionalidad y con esa representación se reconstruye la salida. Los *autoencoders* son herramientas de aprendizaje supervisado, en la que no se requiere especificar la salida deseada, pero si tienen una salida

objetivo, la propia entrada, por lo que se pueden considerar también como herramientas de aprendizaje supervisado. El objetivo de un *autoencoder* es aprender a reducir la dimensionalidad de un grupo de datos, para ignorar el ruido de la señal de entrada y aprender cuáles son sus características definitorias.

Las partes fundamentales del *autoencoder* son dos: un codificador, compuesto de la capa de entrada y la capa oculta, que transforma la entrada (xi) al espacio dimensional $h(xi)$, y un decodificador que realiza la operación inversa, reconstruye la entrada en el espacio de la salida $(\overline{x_i})$ a partir la representación de la entrada (figura 10). Para realizar esta operación se necesita un método de reducción de dimensionalidad de la entrada (codificación), un método para descomprimir la entrada en la salida (decodificación), y una función de pérdida para evaluar la compresión y comparar la salida con el valor deseado.

La aplicación más tradicional de los *autoencoders* es la reducción de dimensionalidad y el aprendizaje de características de conjuntos de datos para el reconocimiento de imágenes, pero en la última década se han estado utilizando para desarrollar potentes redes neuronales de múltiples capas ocultas capaces de procesar gran cantidad de datos. Esto es el conocido aprendizaje profundo o *deep learning.*

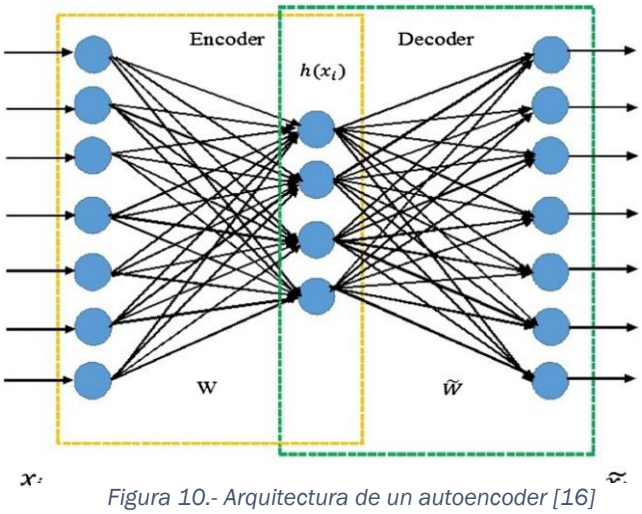

Figura 10.- Arquitectura de un autoencoder [16]

La construcción de redes neuronales con múltiples capas no es fácil. El problema principal se encuentra en la disminución de los gradientes con cada capa oculta, lo que dificulta el seguimiento del error [16]. Uno de los métodos con los que se puede evitar esto y mejores resultados proporciona es el encadenamiento de *autoencoders.* Con este método, se inicializan los

parámetros de la red (pesos y *bias*) para empezar el entrenamiento con los valores más cercanos a los óptimos globales. Esto se conoce como preentrenamiento.

Tras esto, se procede a afinar los resultados mediante algoritmos de entrenamiento tradicionales. La estructura de una red de *autoencoders* encadenados (RAE) se puede ver en la figura 11. Es una red con cuatro capas ocultas (con 6, 5, 4, y 3 neuronas respectivamente), 21 señales de entrada y una señal de salida. Las capas ocultas han sido obtenidas de sendos *autoencoders* en los que no se reduce la dimensionalidad de la entrada, si no que su objetivo es solo aprender a comprimir la entrada y proporcionar esta información a la RAE. En la parte superior se puede ver la estructura de la red de *deep learning*, y en la parte inferior se muestra la estructura de los cinco *autoencoders* con los que se ha construido la red. Estos *autoencoders* tienen una estructura de sus capas (señales de entrada-(neuronas en la capa oculta)-neuronas en la capa de salida): 21-(6)-21, 6-(5)-6, 5-(4)-5, 4-(3)-4, 3-(1)-3. El primer *autoencoder* coge la entrada de la RAE de 21 señales y lo usa para reconstruirlo, comprimiéndolo antes en un espacio de 6 neuronas. Esta capa oculta proporciona una salida con información sobre cómo reducir la dimensión de la entrada, y sirve para proporcionar al siguiente *autoencoder* con una señal de entrada. Básicamente, la salida de la capa oculta del *autoencoder* i-1 se utiliza en el *autoencoder* i como entrada. Se repite el proceso hasta llegar al número de neuronas en la salida deseado. Con cada *autoencoder* entrenado, se obtienen unos pesos (w^i) y *bias* iniciales que se usan como primera aproximación en la RAE. Acabado el preentrenamiento, se procede a usar algoritmos típicos de redes con una única capa oculta, como puede ser la retropopagación del error para hacer un ajuste fino a la red global [16].

Después de la capa oculta se encuentra la capa de salida, que en este caso se denomina capa de clasificación. Es en esta capa donde se produce la clasificación de los datos comprimidos provenientes de la red de aprendizaje profundo. A diferencia de las otras capas de la red, esta se debe entrenar de manera supervisada, suministrando las clases deseadas para la clasificación.

Esta capa puede ser una red de propagación hacia delante estándar o también puede ser de tipo *softmax*, que es una función de transferencia basada en un modelo de regresión que normaliza las entradas de forma exponencial. Siendo *z* las *K* observaciones de entrada:

$$softmax = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, i = 1, \dots, K$$

Las RAE proporcionan ventajas sobre las redes con una única capa oculta en predicciones más precisas con mayores cantidades de datos. Pero su mayor

ventaja, la capacidad de procesamiento de gran cantidad de datos, es también su mayor inconveniente, pues para entrenar correctamente una red así se debe disponer de gran cantidad de datos y de un tiempo de computación considerable. Además, estas técnicas no tienen un trasfondo teórico desarrollado, por lo que elegir parámetros correctos y algoritmos de entrenamiento se apoya en la heurística más que en las matemáticas. [17].

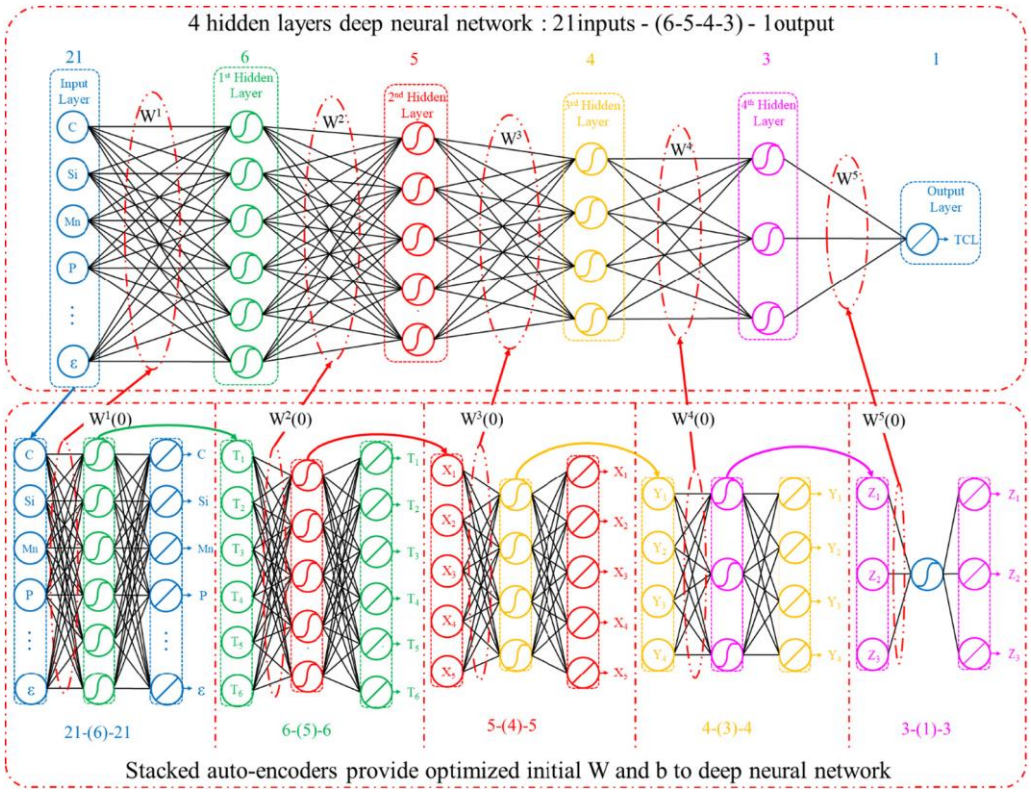

Figura 11.- Esquema del proceso de preentrenamiento mediante encadenamiento de autoencoders [16].

3.3. Árboles de decisión. Bosque aleatorio

Un árbol de decisión es un mapa de los posibles resultados de una serie de decisiones correlacionadas. El aprendizaje por arboles de decisión es un método de aprendizaje automático supervisado usado en el modelado predictivo. Si las salidas deseadas son discretas, los árboles son de clasificación, y si son continuas los árboles son de regresión.

Los árboles están compuestos de nodos, ramas y hojas. Cada nodo representa una característica o atributo, cada rama una decisión o regla y cada hoja un resultado. Básicamente, todos usan un proceso de división en capas, en el que distribuyen a los datos en dos o más grupos, siguiendo unas normas, para agrupar datos similares (homogeneidad) en grupos distintos unos de otros (heterogeneidad). Esta división puede ser binaria, donde desde cada nodo

salen dos ramas, o presentar varias opciones de agrupamiento, con múltiples ramas por nodo.

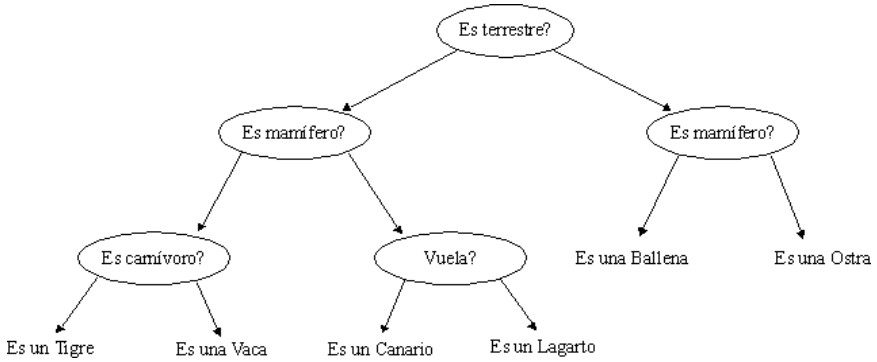

Figura 12.- Ejemplo de un árbol de decisión binario. [24]

La profundidad del árbol es la distancia entre el primer nodo, o nodo raíz, y la última hoja. Una profundidad muy alta permite toma de decisiones complejas, pero un árbol muy profundo es propenso al sobreajuste, lo que significa que ha aprendido, junto al patrón subyacente de los datos, el ruido y aleatoriedad del conjunto de datos, de manera que tiene problemas ajustándose a nuevos datos. Para evitar esto, se define un "hiperparámetro", esto es, un valor o rango de valores usado para controlar el proceso, mediante el cual se excluirá ramas demasiado específicas. Esto se conoce como "poda", y se puede realizar antes o después del entrenamiento.

Existen varios tipos de algoritmos para la creación del árbol, que influyen en su estructura, en el número de ramas que salen de cada nodo. A continuación, se hará un repaso breve de cada uno de ellos [18].

El algoritmo **CHAID** (Chi-squared Automatic Interaction Detection) es uno de los más antiguos, y usa una división por capas múltiple. Para tomar la decisión de la división, se usa la prueba de Chi cuadrado, para comprobar el grado de relación entre las variables. Debido a la multiplicidad de sus capas, los árboles suelen ser demasiado anchos, por lo que no interpretan de manera fiel condiciones reales, además de suponer un tiempo de procesamiento excesivo

El algoritmo **CART** (Classification And Regression Tree) produce árboles binarios, de clasificación o de regresión. Procesa los datos sin necesidad de pretratamiento, y puede usar la misma variable en distintas partes del árbol, lo que puede ayudar a descubrir interdependencias entre conjuntos de variables. Este algoritmo no tiene una medida interna para el rendimiento, y depende de test externos con datos nuevos para comprobar la fidelidad del ajuste. Si es buena, el algoritmo procede a crear otro árbol para intentar mejorar el resultado.

El algoritmo **ID3** (Iterative Dichotomiser 3) se usa para producir árboles de clasificación, pues el algoritmo es incapaz de procesar atributos numéricos. Divide las características de los datos (dicotomiza) en dos grupos para encontrar los atributos dominantes. Usa una métrica llamada ganancia de información, que proporciona el nivel de aleatoriedad que se ha reducido al procesar los datos. Mide cuanta información un atributo nos da de una clase de datos. El atributo que tenga asociado una ganancia de información mayor es el elegido para el nodo del árbol.

El algoritmo **C4.5** es una mejora del ID3 en respecto a que puede ser usado para crear árboles de regresión además de los de clasificación. Es similar en funcionamiento al ID3, usando en vez de la ganancia de información la ratio de ganancia en el proceso de división. Este concepto es una mejora del concepto de ganancia de información, en el que se tiene una cuenta el número y tamaño de las ramas al elegir un atributo. Otra ventaja de este algoritmo respecto al ID3 es que puede "podar" el árbol, calculando el error en cada nodo, y sustituyéndolo por una hoja si el error es muy bajo.

Para solucionar los problemas de sobreajuste y acelerar el procesamiento de datos, los árboles se suelen usar ensamblados unos con otros. Combinando árboles con el mismo algoritmo mejoran el rendimiento respecto a un único árbol. Un conjunto de árboles ensamblados se denomina bosque. Existen dos meta-algoritmos (creados para mejorar procedimientos heurísticos muy generales con alto rendimiento) para ensamblar los árboles, potenciación o *boosting y* empaquetamiento o *bagging*.

La potenciación o *boosting* construye secuencialmente árboles de decisión y a cada paso intenta reducir el error del árbol anterior. Después de cada etapa del entrenamiento, los pesos se redistribuyen basados en los errores calculados, de forma que los árboles con mayor tasa de error tienen asignados pesos más altos que los árboles que no dan problemas, de manera que, en el siguiente paso del entrenamiento, el algoritmo se centrará en ellos y tratará de mejorar la precisión. Hay muchas técnicas para potenciar un bosque, entre ellas se encuentra la potenciación de gradiente, muy usada en el campo de la clasificación de datos.

El empaquetamiento o *bagging* es eficaz en reducir la varianza de un bosque. Para ello, se crean, en paralelo, diferentes subgrupos de los datos de entrenamiento, de forma que cualquier observación tenga la misma probabilidad de aparecer en un nuevo subgrupo. A continuación, cada subgrupo se usa para entrenar los distintos árboles. Para finalizar, se combinan los resultados, mediante una media, si se trata de un problema de regresión, o con moda, si es de clasificación.

Los bosques aleatorios es una extensión del empaquetamiento. Tiene la característica de crear árboles poco correlacionados entre sí. Además de realizar el proceso de empaquetamiento, en cada nodo, escogen un conjunto aleatorio de características de los datos. Con esto evitan uno de los problemas más comunes en los bosques creados con empaquetamiento tradicional. Estos escogen todas las características de los datos en cada nodo, por lo que, si alguna tiene mayor afinidad por las variables de salida que las demás, será elegida un número mayor de veces, lo que desemboca en árboles correlacionados entre sí, con prejuicios respecto a las variables de entrada.

La ventaja de los bosques aleatorio respecto a otras técnicas de empaquetamiento o árboles individuales es la escasa correlación entre árboles. Los errores de los árboles son individuales y no afectan a otros del mismo bosque. Sus ventajas son el buen manejo de grandes cantidades de variables semi continuas gracias a la selección de características aleatoria, el poco efecto que el ruido tiene en ellos, y que transformaciones monótonas de las variables de entrada no implican una variación en las de salida. Por ello, los bosques aleatorios son muy efectivos en el campo de la clasificación de datos.

3.4. Visualización de resultados. Matriz de confusión.

El acabar el entrenamiento de un método de aprendizaje automático supervisado, surge la problemática de la comprobación de resultados, para ver si el ajuste a la situación deseada es correcto o no. Cuando las variables de salida son pocas, esto se puede solucionar comparando directamente los resultados. Con números altos de variables de salida, es conveniente usar herramientas de visualización de datos. Una de estas herramientas es la matriz de confusión, muy usada en problemas de clasificación con variables categóricas. Toma su nombre del hecho que con ella se puede comprobar fácilmente si el algoritmo está “confundiendo” las clases, es decir, si las está clasificando correctamente.

En general todas las matrices se organizan de la misma forma. En la figura 13 se tiene una matriz generada por el software MATLAB™. Cada fila representa la clase predicha, y las columnas se corresponden con las clases deseadas. Las celdas diagonales son observaciones correctamente clasificadas, y las que no se encuentran en esta línea, no están bien clasificadas. La columna a la derecha de la matriz muestra unas métricas que son el porcentaje de las observaciones predichas por el algoritmo que pertenecen a cada clase y que están correctamente (precisión) e incorrectamente (tasa de descubrimientos falsos). La fila de abajo muestras métricas similares, respecto a las salidas deseadas; las clasificadas

correctamente (tasa de verdaderos positivos) e incorrectamente (tasa de falsos negativos). La celda de abajo a la derecha es la más importante, pues muestra la tasa total de clasificaciones correctas e incorrectas.

210 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**4** 0.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**2** 0.1%	**0** 0.0%	97.2% 2.8%
0 0.0%	**210** 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**4** 0.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	98.1% 1.9%
0 0.0%	**0** 0.0%	**209** 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**14** 0.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	93.7% 6.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**208** 9.0%	**0** 0.0%	**0** 0.0%	**6** 0.3%	**2** 0.1%	**1** 0.0%	**5** 0.2%	**0** 0.0%	93.7% 6.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**210** 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**1** 0.0%	**0** 0.0%	99.1% 0.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**153** 6.6%	**0** 0.0%	**0** 0.0%	**33** 1.4%	**39** 1.7%	**0** 0.0%	68.0% 32.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**14** 0.6%	**169** 7.3%	**31** 1.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	78.6% 21.4%
0 0.0%	**0** 0.0%	**1** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**10** 0.4%	**161** 7.0%	**0** 0.0%	**0** 0.0%	**2** 0.1%	92.0% 8.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**34** 1.5%	**22** 1.0%	**0** 0.0%	**125** 5.4%	**62** 2.7%	**0** 0.0%	51.4% 48.6%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**3** 0.1%	**0** 0.0%	**50** 2.2%	**101** 4.4%	**0** 0.0%	65.2% 34.8%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**2** 0.1%	**0** 0.0%	**0** 0.0%	**208** 9.0%	99.0% 1.0%
100% 0.0%	100% 0.0%	99.5% 0.5%	99.0% 1.0%	100% 0.0%	72.9% 27.1%	80.5% 19.5%	76.7% 23.3%	59.5% 40.5%	48.1% 51.9%	99.0% 1.0%	**85.0%** **15.0%**

Figura 13.- Matriz de confusión para un bosque aleatorio con una tasa de 85.0% clasificaciones totales correctas.

CAPITULO III: PLANTA DE ESTUDIO

1. Descripción

En la década de 1980, surgió un interés académico por el desarrollo de un problema real que sirviera de referencia común en el control de procesos químicos. En 1990, la compañía química Eastman y la universidad de Tennessee colaboraron para desarrollar un modelo de planta que sirviera como estándar en el estudio del control de procesos multivariables. Se desarrolló así la planta Tennessee Eastman (TEP), basada en un proceso real de la compañía Eastman, obteniendo así un modelo no lineal considerablemente complejo de un problema multivariable (figura 14). Su utilidad radica en la capacidad de realizar estudios comparativos y de validación de algoritmos, por ejemplo. en el diagnóstico de anomalías en procesos [19].

Se tienen ocho compuestos: cuatro reactivos (A, C, D, E), dos productos (G, H), un subproducto (F) y un inerte (B). Las reacciones que tienen lugar son las siguientes:

- Formación de los productos

$$A_{(g)} + C_{(g)} \rightarrow G_{(l)}$$

$$A_{(g)} + C_{(g)} + E_{(g)} \rightarrow H_{(l)}$$

- Formación de los subproductos

$$A_{(g)} + E_{(g)} \rightarrow F_{(l)}$$

$$3D_{(g)} \rightarrow 2F_{(l)}$$

Las reacciones son exotérmicas e irreversibles, siendo su velocidad función de la temperatura, a través de la ecuación de Arrhenius. La reacción para formar G tiene una energía de activación mayor, por lo que es más sensible a la temperatura. Las reacciones son aproximadamente de primer orden respecto a los reactivos.

La planta consta principalmente de cinco operaciones unitarias: una reacción, una condensación, una separación vapor-líquido, una compresión, y un stripping. Además, hay elementos de control repartidos por todo el proceso: válvulas, indicadores, analizadores, etc.

El proceso comienza con la introducción de los reactivos al reactor, saliendo de este productos y reactivos no reaccionados. Un catalizador no volátil está presente en la fase líquida, y no abandona el reactor en ningún momento. El reactor cuenta con un sistema de refrigeración para eliminar el calor de la reacción, acelerándola. La salida del reactor se lleva a un condensador y se introduce a un separador liquido-vapor. Los no condensables se recirculan a través de un compresor centrífugo, para ser mezclados con la

alimentación de A. D y E. Los condensables se trasladan a una columna de stripping, que es alimentada con el reactivo C. Con esto se eliminan los reactivos de la corriente de productos del reactor. Por la base de la columna de stripping salen los productos G y H y se separan aguas abajo en una sección de refinamiento no incluida en este caso de estudio. Los inertes y subproductos son purgados en la corriente de vapor del separador vapor-líquido.

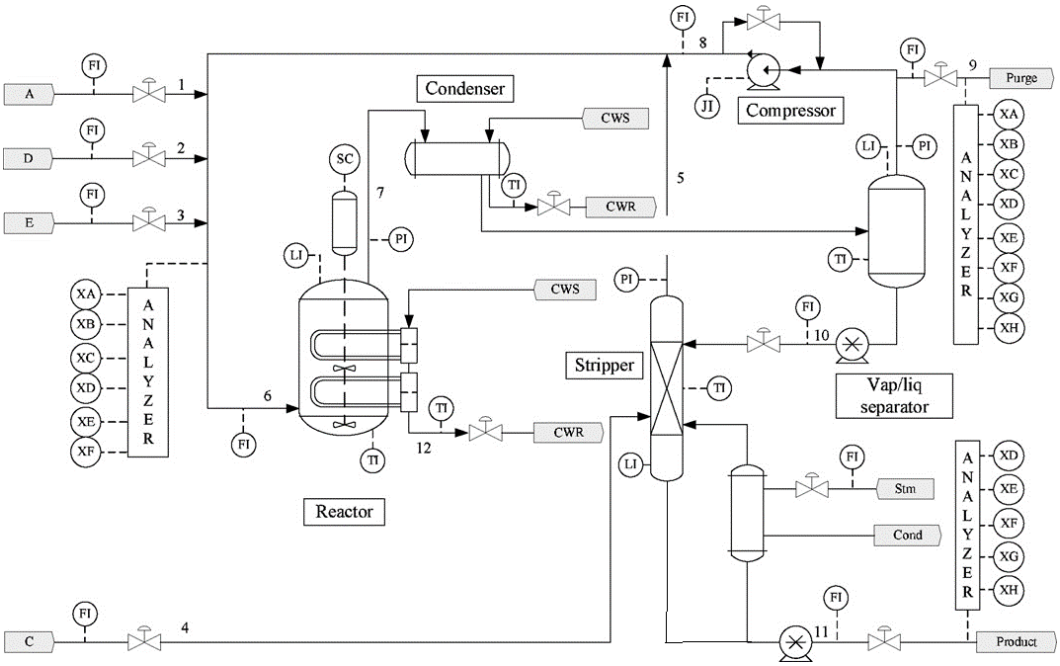

Figura 14. Proceso Tennessee Eastman [20].

2. Datos de la planta

El modelo del proceso cuenta con 12 variables manipuladas (tabla 1) y 41 variables medidas, de las cuales las primeras 22 son variables del proceso (tabla 2), y los 19 restantes son variables procedentes del muestreo de la planta (tabla 3). En la planta existe la posibilidad de que ocurran 21 fallos distintos (tabla 4). En total, se cuenta con 53 variables de proceso, y con 21 fallos que hay que detectar.

Tabla 1. Variables manipuladas del proceso

Variable	Numero de la variable	Unidades
Flujo de alimentación D (corriente 2)	XMV(1)	kg h^{-1}
Flujo de alimentación E (corriente 3)	XMV(2)	kg h^{-1}
Flujo de alimentación A (corriente 1)	XMV(3)	kscmh

Flujo de alimentación A y C (corriente 4)	XMV(4)	kscmh
Válvula de recirculación del compresor	XMV(5)	%
Válvula de purga (corriente 9)	XMV(6)	%
Flujo de líquido del separador LV (corriente 10)	XMV(7)	$m^3\ h^{-1}$
Flujo de líquido de la columna de stripping (corriente 11)	XMV(8)	$m^3\ h^{-1}$
Válvula de vapor de la columna de stripping	XMV(9)	%
Flujo de agua de refrigeración del reactor	XMV(10)	$m^3\ h^{-1}$
Flujo de agua en el condensador	XMV(11)	$m^3\ h^{-1}$
Velocidad del agitador del reactor	XMV(12)	rpm

Tabla 2. Variables medidas del proceso

Variable	Numero de la variable	Unidades
Flujo de alimentación A (corriente 1)	XMEAS(1)	kscmh
Flujo de alimentación D (corriente 2)	XMEAS(2)	kscmh
Flujo de alimentación E (corriente 3)	XMEAS(3)	kscmh
Flujo de alimentación A y C (corriente 4)	XMEAS(4)	kscmh
Flujo de recirculación (corriente 8)	XMEAS(5)	kscmh
Flujo alimentación reactor (corriente 6)	XMEAS(6)	kscmh
Presión del reactor	XMEAS(7)	kPa
Nivel del reactor	XMEAS(8)	%
Temperatura del reactor	XMEAS(9)	ºC
Flujo de purga (corriente 9)	XMEAS(10)	kscmh
Temperatura del separador de producto	XMEAS(11)	ºC
Nivel del separador de producto	XMEAS(12)	%
Presión del separador de producto	XMEAS(13)	kPa
Corriente del separador de producto (corriente 10)	XMEAS(14)	m^3h^{-1}
Nivel del stripper	XMEAS(15)	%
Presión del stripper	XMEAS(16)	kPa
Corriente del stripper (corriente 11)	XMEAS(17)	m^3h^{-1}
Temperatura del stripper	XMEAS(18)	ºC
Flujo de vapor del stripper	XMEAS(19)	kgh^{-1}

Potencia del compresor	XMEAS(20)	kW
Temperatura de la salida del agua de refrigeración del reactor	XMEAS(21)	ºC
Temperatura de la salida del agua de refrigeración de separador	XMEAS(22)	ºC

Tabla 3.- Variables medidas muestreadas

Muestreo de la alimentación del reactor (corriente 6)		
Variable	Numero de la variable	Unidades
A	XMEAS(23)	%mol
B	XMEAS(24)	%mol
C	XMEAS(25)	%mol
D	XMEAS(26)	%mol
E	XMEAS(27)	%mol
F	XMEAS(28)	%mol
Muestreo del gas de purga (corriente 9)		
Variable	Numero de la variable	Unidades
A	XMEAS(29)	%mol
B	XMEAS(30)	%mol
C	XMEAS(31)	%mol
D	XMEAS(32)	%mol
E	XMEAS(33)	%mol
F	XMEAS(34)	%mol
G	XMEAS(35)	%mol
H	XMEAS(36)	%mol
Muestreo del producto final (corriente 11)		
Variable	Numero de la variable	Unidades
D	XMEAS(37)	%mol
E	XMEAS(38)	%mol
F	XMEAS(39)	%mol
G	XMEAS(40)	%mol
H	XMEAS(41)	%mol
F	XMEAS(42)	%mol

Tabla 4. Fallos posibles del proceso

Variable	Número del fallo	Tipo
Ratio A/C de alimentación, con B constante (corriente 4)	IDV(1)	Salto
Composición de B, con ratio A/C constante (corriente 4)	IDV(2)	Salto
Temperatura de la alimentación de D (corriente 2)	IDV(3)	Salto
Temperatura de entrada del agua de refrigeración del reactor	IDV(4)	Salto
Temperatura de entrada del agua de condensación	IDV(5)	Salto
Pérdida en la alimentación de A (corriente 1)	IDV(6)	Salto
Perdida de presión en la alimentación de C (corriente 4)	IDV(7)	Salto
Composición de A, B, C (corriente 4)	IDV(8)	Variación aleatoria
Temperatura de la alimentación de D (corriente 2)	IDV(9)	Variación aleatoria
Temperatura de alimentación de C (corriente 4)	IDV(10)	Variación aleatoria
Temperatura de entrada del agua de refrigeración del reactor	IDV(11)	Variación aleatoria
Temperatura de entrada del agua de condensación	IDV(12)	Variación aleatoria
Cinética de reacción	IDV(13)	Variación lenta
Apertura de la válvula del agua de refrigeración del reactor	IDV(14)	Sin variación, constante en un valor
Apertura de la válvula del agua de condensación	IDV(15)	Sin variación, constante en un valor
Desconocido	IDV(16)	Desconocido
Desconocido	IDV(17)	Desconocido
Desconocido	IDV(18)	Desconocido
Desconocido	IDV(19)	Desconocido
Desconocido	IDV(20)	Desconocido
Desconocido	IDV(21)	Desconocido

El proceso tiene más variables manipuladas que las necesarias para controlar la calidad y cantidad del producto. Esto es así para permitir el estudio de la optimización en este aspecto [20]. Los fallos representan posibles variaciones en el comportamiento de diferentes variables en la planta, y pueden ser saltos en los valores de las variables, o pueden ser variaciones aleatorias. Un tipo especial de fallo es la apertura de las válvulas, que se mantiene constante. Los fallos 16 a 21 se desconoce la variable a la que influyen principalmente, pero aun así causan variaciones indeseadas en la planta. Especialmente relevantes son los fallos 3, 9 y 15, Análisis previos de estos fallos indican que las variables no sufren cambios significativos respecto al comportamiento normal, por lo que son difíciles de detectar con métodos estadísticos normales. [21].

CAPITULO IV: APLICACIÓN DESARROLLADA.

1. Conocimientos previos

Para facilitar la experimentación, se cuenta con ciertos conocimientos extraídos de trabajos previos con estas técnicas. En primer lugar, se sabe que una red de propagación hacia delante con una sola capa oculta con 10 neuronas tiene problemas para detectar más de cuatro fallos [22]. En segundo lugar, los datos necesarios para la experimentación no se generan, sino que se obtienen de la de la bibliografía, ya que son datos de un *benchmark* utilizados para comparar diferentes métodos de control y monitorización sobre una misma planta. En simulaciones previas [10], se han tomado datos de la planta, midiéndose cada una de las 52 variables, realizando 22 simulaciones diferentes: una para obtener datos de comportamiento normal, y las 21 restantes correspondientes al comportamiento normal de la planta con la presencia de un único fallo. Cada simulación se recoge en sendos ficheros de datos genéricos, y se pueden obtener desde la página web correspondiente [23]. El conjunto de datos está dividido en dos tipos de ficheros diferentes: ficheros de entrenamiento, con 960 observaciones, las 160 primeras correspondientes a comportamiento normal, para crear la herramienta de aprendizaje automático; y ficheros de validación, con 480 observaciones, para comprobar la calidad de la herramienta desarrollada. Como ya se ha comentado, se suprimirá de la experimentación los fallos 3, 9 y 15 por ser difíciles de detectar debido a las pequeñas variaciones que sufren respecto al comportamiento normal. El fallo 6 también se omitirá porque solo se dispone del fichero de validación, sin el de entrenamiento. Por lo tanto, se trabajará como máximo con 17 fallos.

Se trabajará en el software MATLAB™. Para cada caso de estudio se utilizarán dos ficheros; uno para crear y probar la técnica de aprendizaje automático, usando los ficheros de datos de entrenamiento, y otro para validarlo, usando los ficheros de datos de validación. Los datos de entrenamiento se dividirán de forma que, las observaciones 161 a la 750 (590 observaciones), se utilizarán para crear la herramienta, y las observaciones 751 a la 960 (210 observaciones) serán usadas para probar su funcionamiento, es decir, para probar la red con datos diferentes a los usados en el entrenamiento, y ver su capacidad de generalización.

2. Detección de fallos mediante análisis de componentes principales.

Para desarrollar el PCA, primero hay que desarrollar la aplicación fuera de línea, y después aplicar el método calculado con datos de fallo en línea. Se ha seguido el procedimiento descrito en el apartado teórico.

2.1. Fuera de línea

En la aplicación fuera de línea, crea la matriz de datos de comportamiento normal, X, de dimensiones 590x52. Es este espacio, R^{590x52}, al que se le reducirá las dimensiones mediante el PCA.

Se normalizan los datos de entrada a media cero y varianza unidad, para que todos los datos tengan la misma importancia. Se crea la matriz de correlación de X, A, y se extraen los valores singulares de A en orden decreciente, con lo que se obtienen las matrices Λ, donde se almacenan los valores propios, y V, donde se almacenan los vectores propios o *scores* en las columnas. Se trabaja con una variabilidad deseada del 90%, y un porcentaje de falsas alarmas del 0.01 %. Con esto, se calcula el número de componentes principales, a, usando el test de porcentaje de varianza.

En este caso, se obtienen 31 componentes principales. A continuación, se define la matriz de carga, P, reteniendo los 31 vectores de carga correspondientes a los 31 valores singulares más grandes. Esto se hace a partir de la matriz V, en la que se seleccionan solo los 31 primeros elementos de las columnas, que almacenaban los vectores propios. Una vez obtenida P, se proyecta desde el espacio R^{52} a uno tal que R^{31} (ecuación 3), y se obtiene el conjunto de datos transformados o componentes principales, T.

Con estos resultados se procede a calcular los estadísticos T^2 y Q. El estadístico T^2 se calcula aplicando directamente la ecuación 7, tras lo cual se procede a definir el umbral, usando la ecuación 8. El estadístico Q se calcula aplicando la ecuación 10 para hallar los vectores de residuos, y a continuación se aplica la ecuación 9 para hallar el estadístico. El umbral se calculó con la distribución de Chi-cuadrado (ecuación 11).

2.2. En línea

Para comprobar si el método creado fuera de línea para la situación normal de la planta es capaz de detectar los fallos de la planta, se prueba en

línea. Para ello, se cogen datos de la planta en las condiciones actuales. Con estos datos, y el modelo PCA calculado fuera de línea, se calculan las estadísticas T^2 y Q, y se comparan con sus umbrales. Si alguna de las estadísticas supera su umbral, se detecta un fallo. Para evitar falsas alarmas, normalmente se necesita que las estadísticas superen su umbral un numero consecutivo de veces antes de lanzar la alarma de fallos. Este número no debe ser grande para que no se retrase mucho la detección, pero no debe ser pequeño para evitar falsas alarmas.

Los resultados de la detección del fallo 1 y 7 están en las figuras 15 y 16. La línea roja representa el umbral del estadístico, y se puede ver como las primeras 160 observaciones no superan el umbral, pues corresponden al comportamiento normal, y el resto de las observaciones hasta la 960 están por encima, ya que son datos de fallo. Una vez detectado el fallo es necesario identificarlo, lo que incluye localizarlo, identificar las variables responsables del mismo, y/o diagnosticar que fallo ha ocurrido. Una primera aproximación se hará con el diagrama de contribuciones, explicado en el capítulo 2, sección 2.3. Se ha realizado un diagrama mediante el cálculo de los residuos (ecuación 10) para entender cuál son las variables que más influencia tienen para causar los fallos. Como ejemplo, el fallo 1 (figura 17) y el fallo 7 (figura 18). Se puede ver que el fallo 1, correspondiente a una anomalía en la ratio A/C de alimentación, con la composición de B constante, se ve más afectado por las variables 16, la presión en el stripper, y la 20, la potencia del compresor. Este resultado se puede explicar observando el diagrama de proceso (figura 14): el reactivo C se alimenta directamente al stripper, por lo que una variación de su presión afecta a la alimentación, al aumentar o disminuir la cantidad que entra a la torre, y lo mismo ocurre con el compresor, que recircula los incondensables directamente a la línea de alimentación de A. El fallo 7 corresponde a una pérdida de presión en la alimentación C, y se ve más afectado por la variable 16, que es la presión en el stripper, y como ya se ha explicado, C se alimenta directamente al stripper, por lo que una variación de su presión afecta a la alimentación.

Se han hecho pruebas con todos los fallos, y todos se han conseguido detectar siguiendo este método. El tiempo de detección ha sido corto, como se ve en las figuras, a partir de los 160 primeros datos de comportamiento normal, el sistema detecta casi inmediatamente el fallo.

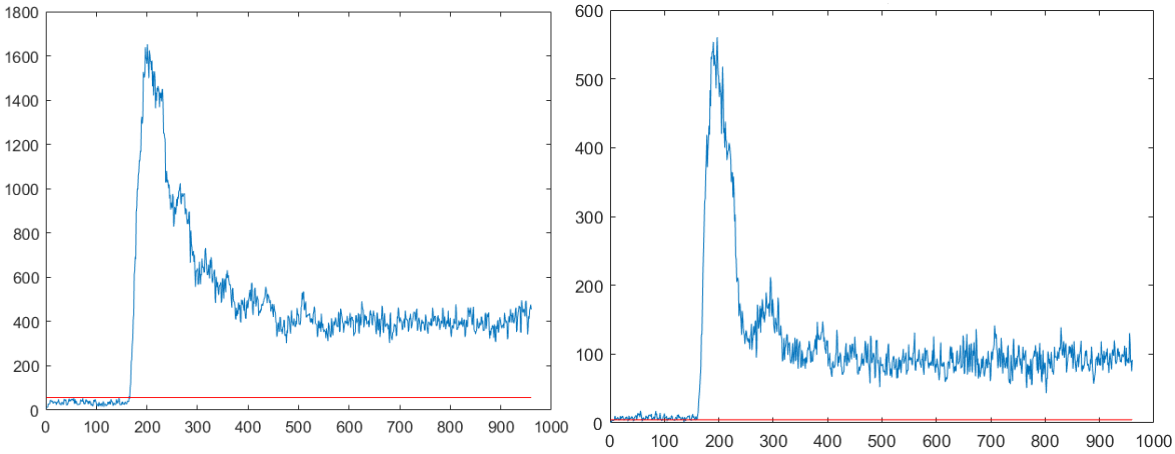

Figura 15.- Resultados de la detección del fallo 1. A la izquierda, el T2 y su umbral; a la derecha, Q y su umbral.

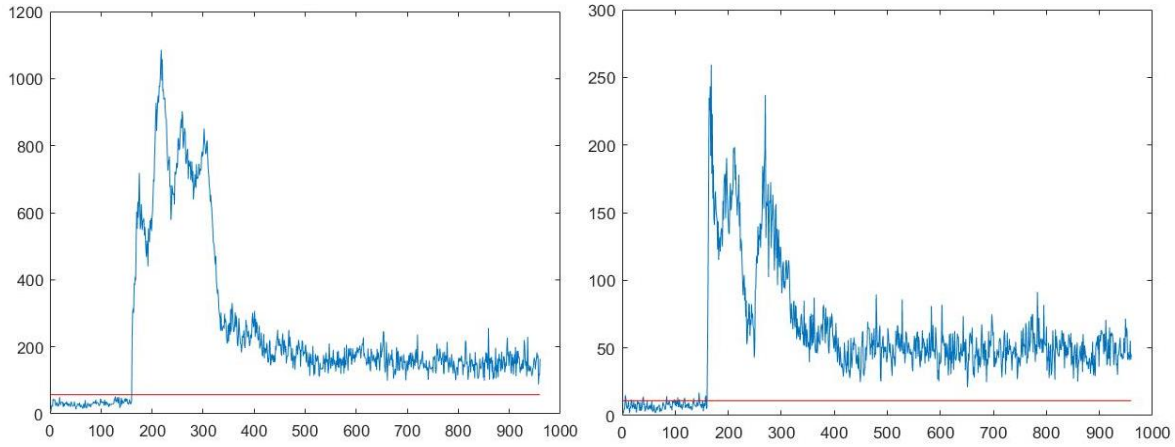

Figura 16.- Resultados de la detección del fallo 7. A la izquierda, el T2 y su umbral; a la derecha, Q y su umbral.

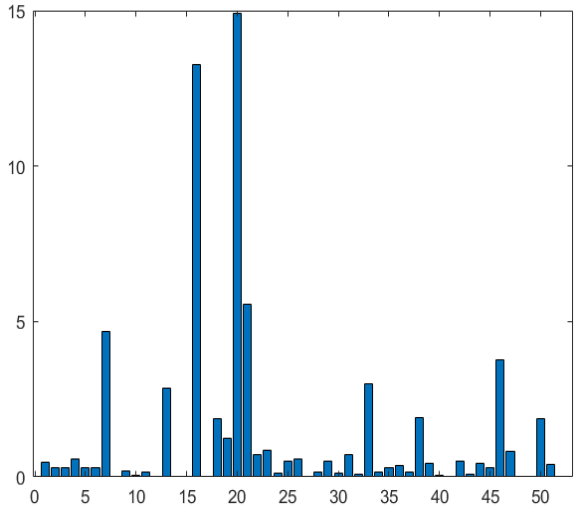

Figura 17.- Diagrama de contribuciones del fallo 1.

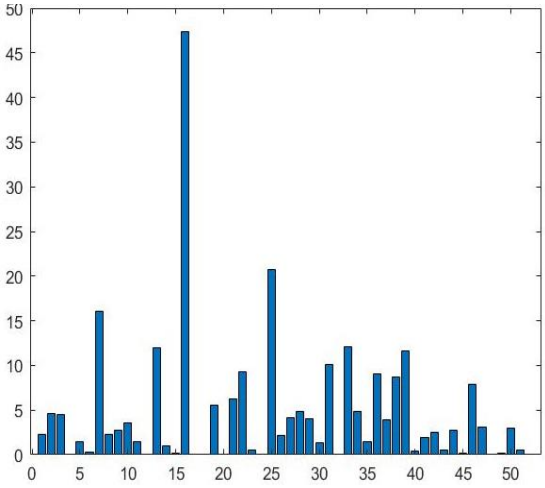

Figura 18.- Diagrama de contribuciones del fallo 7.

3. Diagnóstico de fallos mediante técnicas de aprendizaje automático.

Como ya se ha explicado, una vez detectado un fallo en la planta, es necesario diagnosticar el fallo que ha ocurrido. Para esto hay varias técnicas, y a partir de aquí, se va a trabajar con métodos de clasificación. Es decir, se va a intentar clasificar si los datos actuales de la planta que presentan un fallo, detectado con PCA, pertenecen a una clase determinada (fallo 1, fallo 2…). Para realizar se va a utilizar técnicas de aprendizaje automático como redes neuronales, redes con aprendizaje profundo, usando *autoencoders* encadenados y arboles de decisión ensamblados.

3.1. Red de propagación hacia delante.

A continuación, se describe el uso de las redes de propagación hacia adelante para el diagnóstico de fallos.

3.1.1. Una capa oculta

Se comenzó la experimentación con las técnicas de aprendizaje automático trabajando con una red de propagación hacia delante con una sola capa oculta con 10 neuronas para detectar cuatro fallos, replicando la experiencia mencionada previamente. A continuación, se fueron añadiendo más fallos, y se probó con un número mayor de neuronas en cada caso.

3.1.1.1. Cuatro fallos

Se crea una matriz de datos, X, con los datos correspondientes a los cuatro primeros fallos (1, 2, 4 y 5), desde las observaciones 161 a 750 (590 observaciones por cada fallo), de dimensiones 2360x52. Se normaliza entre -1 y 1 para que todas las variables tengan el mismo peso matemático.

A continuación, se debe crear la matriz de salida deseada, Y_d, para establecer la clasificación. En este caso, al querer detectar un suceso binario, se definirá la matriz tal que, si ese fallo pertenece a una clase, la red neuronal devolverá un uno, y si no pertenece a esa clase, devolverá un cero. La forma de hacerlo es con una matriz de unos y ceros escalonados; tiene 4 columnas, una por cada fallo, y 2360 filas, una por cada observación en X. La primera columna corresponderá al fallo 1, por lo tanto, las 590 primeras observaciones valdrán uno, y el resto de la columna valdrá cero. En la segunda columna, correspondiente al fallo 2, las observaciones 591 a 1180 valdrán uno, y el resto cero. Así se repite con el resto de las columnas. De esta forma, cuando la red aprenda a reconocer los fallos, asignará el valor uno a la primera neurona de la capa de salida, y cero a todas las demás neuronas, cuando haya fallo 1; con datos de fallo 2, la única neurona de la capa de salida que valdrá uno será la segunda neurona, y así sucesivamente.

Con los datos de entrada, X, y salida deseados, Y_d, se entrenan varias redes neuronales, donde lo único que se modifica es el número de neuronas en la capa oculta, que se varío desde 10 hasta 30 neuronas. El mejor resultado se obtuvo con 10 neuronas en la capa oculta.

Para comprobar la calidad del entrenamiento, es decir, para probar la red con datos distintos de los usados en el entrenamiento, se usan las 210 observaciones reservadas. Se crea la matriz de datos de entrada para testear, igual que se hizo con los datos de entrenamiento, X_t, y se normaliza cada una con los mismos parámetros con los que se normalizó X. Se aplica la red neuronal a estos datos de entrada, y se guardan los resultados en matrices de salida, Y_t. Para visualizar los resultados, en una misma figura se representan varios gráficos para cada una de las cuatro columnas, correspondiente a un fallo. En la figura 19 se puede ver la salida de la red neuronal cuando se alimenta con los datos de prueba correspondientes al fallo 1. La clasificación es prácticamente perfecta, ya que la primera columna, correspondiente a este fallo, vale uno, y el resto valen cero. Con el resto de los datos de prueba para los distintos fallos ocurre lo mismo, las clasificaciones son correctas.

Ahora, se prueba la red entrenada con datos nuevos cogidos de la planta para ver si es capaz de clasificar correctamente. Para el mismo fallo 1, ahora la clasificación no es del todo correcta, habiendo observaciones que se desvían

del valor que deberían tener, pero aun así es un resultado aceptable. Los resultados se pueden ver en la figura 20.

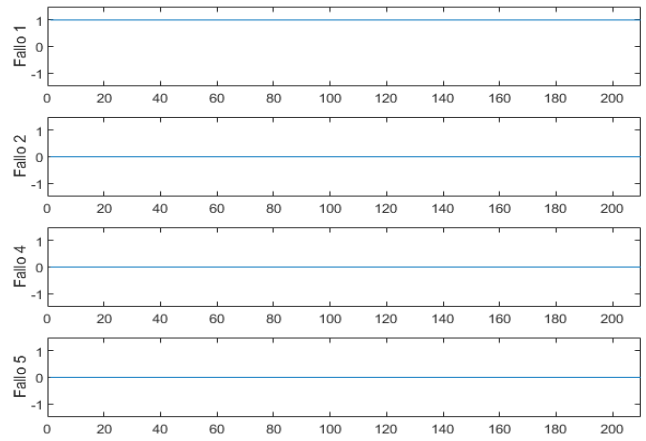

Figura 19.- Salida de la red neuronal para el fallo 1 con los datos X_t.

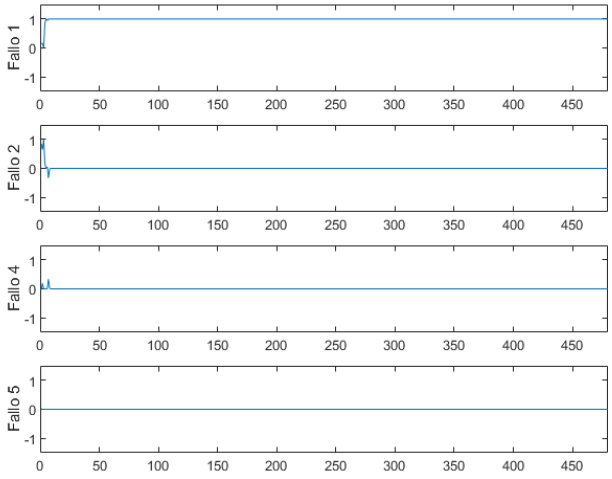

Figura 20.- Salida de la red neuronal para el fallo 1 en la validación.

3.1.1.2. Más de cuatro fallos

El procedimiento es idéntico al descrito para la red neuronal de propagación hacia delante con cuatro fallos, solo que ahora se irá incrementando el número de fallos que queremos clasificar, para comprobar la capacidad de la red. Como era de esperar, una red tan simple tiene problemas con cantidades altas de datos. Añadiendo un fallo más, el fallo 7, con la clasificación empieza a surgir problemas como se puede ver en la figura 21. Cuando la red tiene que procesar más de cinco fallos, aparecen clasificaciones incorrectas (figura 21). Aumentando el número de neuronas de 10 a 30 se mitigan las fluctuaciones (figura 22), pero a costa de incrementar el tiempo de

computación considerablemente y sin obtener resultados que se puedan considerar aceptables

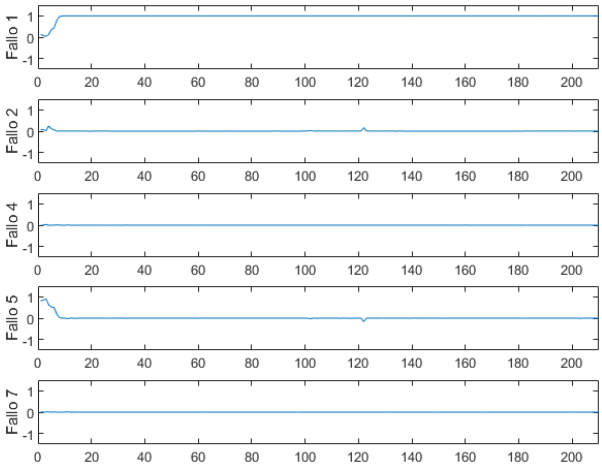

Figura 21.- Salida de la red para el fallo 1 en la validación

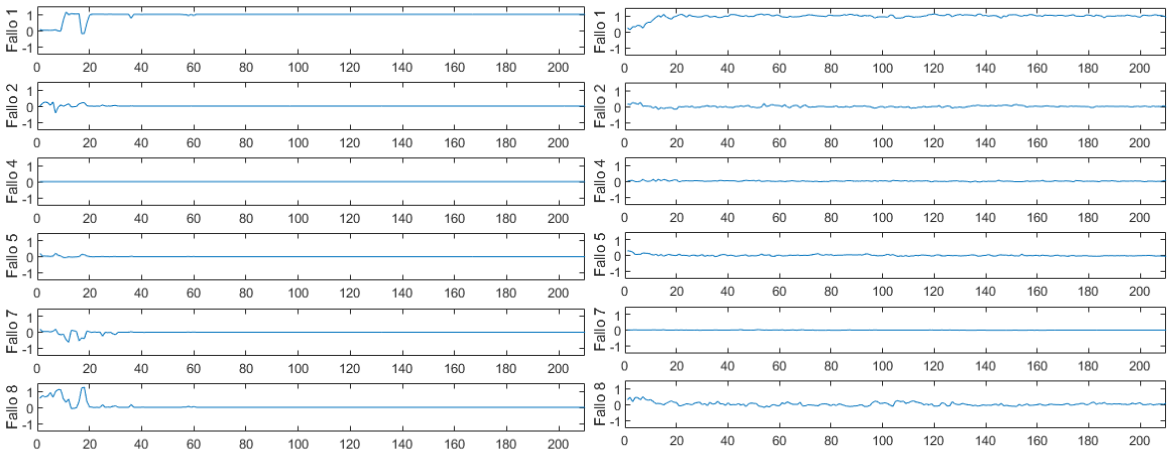

Figura 22.- Salida de la red para el fallo 1 en la validación. A la izquierda, 10 neuronas, a la derecha, 30 neuronas.

Se continuó experimentando con más fallos, pero no se probó con más de siete fallos porque los resultados empezaban a ser previsiblemente malos. En la figura 23 se ve como se probó a añadir fallos nuevos (10 y 11) y eliminar el fallo 8, manteniendo las 30 neuronas, para comprobar si eliminando un fallo que la red tuviera problemas en clasificar influiría en el resultado, y no fue así. Incrementar el número de neuronas no pareció mejorar la clasificación.

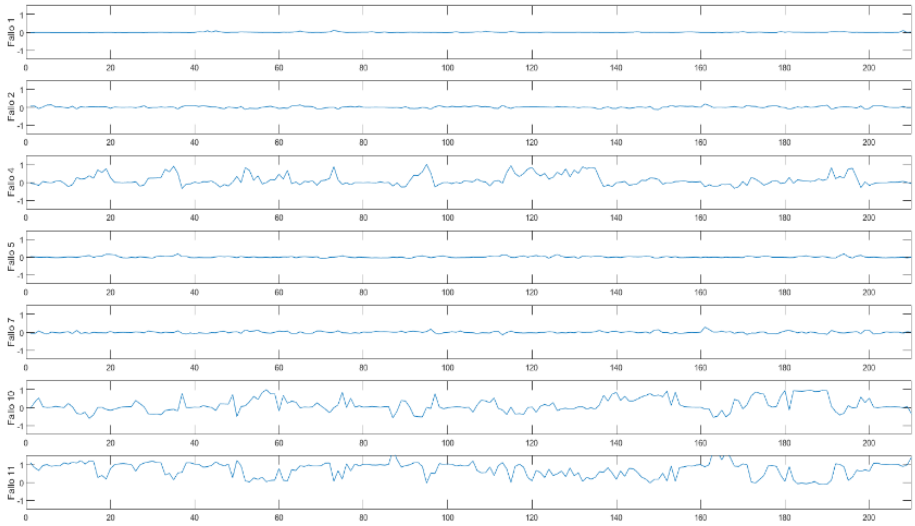

Figura 23.- Salida de la red para el fallo 11 en la validación.

3.1.2. Red multicapa.

Para incrementar la capacidad de la red para procesar datos, se realizaron varias experiencias con redes de propagación hacia delante con más de una capa oculta con un número decreciente de neuronas. La elección del número de neuronas se basa en la prueba y error, teniendo en cuenta que esta técnica es limitada y no admite reducciones de dimensiones bruscas, por lo que los números de neuronas entre capas consecutivas no tendrán que variar mucho. Se harán las pruebas con 11 fallos (del fallo 1 al fallo 14) y con 17 fallos, y se utilizarán redes de hasta cuatro capas ocultas, incluyendo una sola capa otra vez, para comparar.

Existen varios algoritmos de entrenamiento para la red de propagación hacia delante. Por defecto se utiliza un algoritmo, que calcula los pesos y bias utilizando la optimización de *Levenberg-Marquardt*. Es adecuado para una primera aproximación, por su rapidez, pero no es adecuado para una red con un número alto de parámetros, como es este caso. Se utilizará otro algoritmo, que calcula los parámetros usando el método del gradiente conjugado. Este es muchísisimo más rápido que el algoritmo anterior, y da mejores resultados para un número de pesos elevado, además de requerir una cantidad menor de memoria.

El procedimiento es similar al seguido para una sola capa oculta. Ahora se carga un número mayor de datos, se definirán las capas ocultas con las respectivas neuronas, y los resultados se visualizarán con una matriz de confusión. El resto del procedimiento permanece igual.

Se han hecho pruebas para 11 y 17 fallos, con: una sola capa oculta, con 30 neuronas; dos capas ocultas, con 30 y 15 neuronas; tres capas ocultas, con 40, 30 y 25 neuronas; y cuatro capas ocultas, con 40, 30, 25 y 20 neuronas. Los resultados con los datos de validación de estas redes, es decir con datos no vistos durante el entrenamiento, en concreto su matriz de confusión se puede ver en las figuras, 24 a 31.

468 8.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**58** 1.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**16** 0.3%	**0** 0.0%	86.3% 13.7%
0 0.0%	**451** 8.5%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
0 0.0%	**2** 0.0%	**474** 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**100** 1.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	82.3% 17.7%
0 0.0%	**0** 0.0%	**0** 0.0%	**466** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**11** 0.2%	**10** 0.2%	**0** 0.0%	95.7% 4.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**478** 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
7 0.1%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	**288** 5.5%	**89** 1.7%	**61** 1.2%	**22** 0.4%	**187** 3.5%	**0** 0.0%	43.9% 56.1%
4 0.1%	**5** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**3** 0.1%	**29** 0.5%	**15** 0.3%	**4** 0.1%	**3** 0.1%	**0** 0.0%	46.0% 54.0%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**27** 0.5%	**141** 2.7%	**1** 0.0%	**3** 0.1%	**0** 0.0%	81.5% 18.5%
1 0.0%	**15** 0.3%	**2** 0.0%	**10** 0.2%	**2** 0.0%	**42** 0.8%	**239** 4.5%	**79** 1.5%	**401** 7.6%	**78** 1.5%	**0** 0.0%	46.1% 53.9%
0 0.0%	**7** 0.1%	**0** 0.0%	**4** 0.1%	**0** 0.0%	**89** 1.7%	**95** 1.8%	**63** 1.2%	**41** 0.8%	**181** 3.4%	**0** 0.0%	37.7% 62.3%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**21** 0.4%	**0** 0.0%	**2** 0.0%	**480** 9.1%	95.0% 5.0%
97.5% 2.5%	94.0% 6.0%	98.8% 1.2%	97.1% 2.9%	99.6% 0.4%	60.0% 40.0%	6.0% 94.0%	29.4% 70.6%	83.5% 16.5%	37.7% 62.3%	100% 0.0%	**73.0%** **27.0%**

Figura 24.- Matriz de confusión de la salida de la validación de la red con una capa oculta y 11 fallos

468 8.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**114** 2.2%	**0** 0.0%	**0** 0.0%	**21** 0.4%	**25** 0.5%	**0** 0.0%	74.5% 25.5%
0 0.0%	**462** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	99.8% 0.2%
0 0.0%	**2** 0.0%	**479** 9.1%	**1** 0.0%	**3** 0.1%	**14** 0.3%	**8** 0.2%	**158** 3.0%	**6** 0.1%	**1** 0.0%	**65** 1.2%	65.0% 35.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**396** 7.5%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**3** 0.1%	**21** 0.4%	**0** 0.0%	94.3% 5.7%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**474** 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
8 0.2%	**1** 0.0%	**0** 0.0%	**4** 0.1%	**0** 0.0%	**185** 3.5%	**22** 0.4%	**6** 0.1%	**40** 0.8%	**81** 1.5%	**0** 0.0%	53.3% 46.7%
1 0.0%	**0** 0.0%	**0** 0.0%	**4** 0.1%	**0** 0.0%	**1** 0.0%	**13** 0.2%	**3** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	59.1% 40.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**10** 0.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	90.9% 9.1%
3 0.1%	**15** 0.3%	**0** 0.0%	**75** 1.4%	**3** 0.1%	**136** 2.6%	**407** 7.7%	**253** 4.8%	**380** 7.2%	**126** 2.4%	**72** 1.4%	25.9% 74.1%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**30** 0.6%	**29** 0.5%	**13** 0.2%	**29** 0.5%	**225** 4.3%	**3** 0.1%	68.2% 31.8%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**37** 0.7%	**1** 0.0%	**0** 0.0%	**340** 6.4%	89.9% 10.1%
97.5% 2.5%	96.3% 3.7%	99.8% 0.2%	82.5% 17.5%	98.8% 1.2%	38.5% 61.5%	2.7% 97.3%	2.1% 97.9%	79.2% 20.8%	46.9% 53.1%	70.8% 29.2%	**65.0%** **35.0%**

Figura 25.- Matriz de confusión de la salida de la validación de la red con dos capas ocultas y 11 fallos.

468 8.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**58** 1.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**16** 0.3%	**0** 0.0%	86.3% 13.7%
0 0.0%	**451** 8.5%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
0 0.0%	**2** 0.0%	**474** 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**100** 1.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	82.3% 17.7%
0 0.0%	**0** 0.0%	**0** 0.0%	**466** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**11** 0.2%	**10** 0.2%	**0** 0.0%	95.7% 4.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**478** 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
7 0.1%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	**288** 5.5%	**89** 1.7%	**61** 1.2%	**22** 0.4%	**187** 3.5%	**0** 0.0%	43.9% 56.1%
4 0.1%	**5** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**3** 0.1%	**29** 0.5%	**15** 0.3%	**4** 0.1%	**3** 0.1%	**0** 0.0%	46.0% 54.0%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**27** 0.5%	**141** 2.7%	**1** 0.0%	**3** 0.1%	**0** 0.0%	81.5% 18.5%
1 0.0%	**15** 0.3%	**2** 0.0%	**10** 0.2%	**2** 0.0%	**42** 0.8%	**239** 4.5%	**79** 1.5%	**401** 7.6%	**78** 1.5%	**0** 0.0%	46.1% 53.9%
0 0.0%	**7** 0.1%	**0** 0.0%	**4** 0.1%	**0** 0.0%	**89** 1.7%	**95** 1.8%	**63** 1.2%	**41** 0.8%	**181** 3.4%	**0** 0.0%	37.7% 62.3%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**21** 0.4%	**0** 0.0%	**2** 0.0%	**480** 9.1%	95.0% 5.0%
97.5% 2.5%	94.0% 6.0%	98.8% 1.2%	97.1% 2.9%	99.6% 0.4%	60.0% 40.0%	6.0% 94.0%	29.4% 70.6%	83.5% 16.5%	37.7% 62.3%	100% 0.0%	**73.0%** **27.0%**

Figura 26.- Matriz de confusión de la salida de la validación de la red con tres capas ocultas y 11 fallos.

472 8.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**55** 1.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	89.6% 10.4%
0 0.0%	**456** 8.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
0 0.0%	**0** 0.0%	**144** 2.7%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**11** 0.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	92.9% 7.1%
0 0.0%	**0** 0.0%	**0** 0.0%	**451** 8.5%	**2** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	98.9% 1.1%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**457** 8.7%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
4 0.1%	**4** 0.1%	**8** 0.2%	**11** 0.2%	**8** 0.2%	**362** 6.9%	**72** 1.4%	**11** 0.2%	**103** 2.0%	**293** 5.5%	**0** 0.0%	41.3% 58.7%
4 0.1%	**5** 0.1%	**3** 0.1%	**6** 0.1%	**0** 0.0%	**6** 0.1%	**84** 1.6%	**32** 0.6%	**28** 0.5%	**12** 0.2%	**1** 0.0%	46.4% 53.6%
0 0.0%	**10** 0.2%	**306** 5.8%	**1** 0.0%	**2** 0.0%	**7** 0.1%	**59** 1.1%	**284** 5.4%	**6** 0.1%	**5** 0.1%	**3** 0.1%	41.6% 58.4%
0 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**14** 0.3%	**19** 0.4%	**3** 0.1%	**252** 4.8%	**26** 0.5%	**0** 0.0%	80.0% 20.0%
0 0.0%	**5** 0.1%	**18** 0.3%	**10** 0.2%	**10** 0.2%	**36** 0.7%	**245** 4.6%	**84** 1.6%	**88** 1.7%	**140** 2.7%	**0** 0.0%	22.0% 78.0%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**55** 1.0%	**1** 0.0%	**4** 0.1%	**476** 9.0%	88.5% 11.5%
98.3% 1.7%	95.0% 5.0%	30.0% 70.0%	94.0% 6.0%	95.2% 4.8%	75.4% 24.6%	17.5% 82.5%	59.2% 40.8%	52.5% 47.5%	29.2% 70.8%	99.2% 0.8%	**67.8%** **32.2%**

Figura 27.- Matriz de confusión de la salida de la validación de la red con cuatro capas ocultas y 11 fallos.

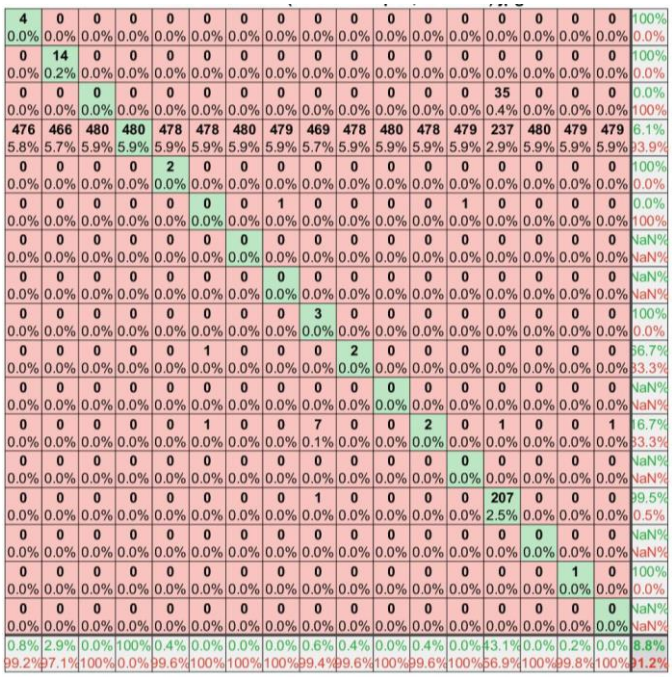

Figura 28.- Matriz de confusión de la salida de la validación de la red con una capa oculta y 17 fallos.

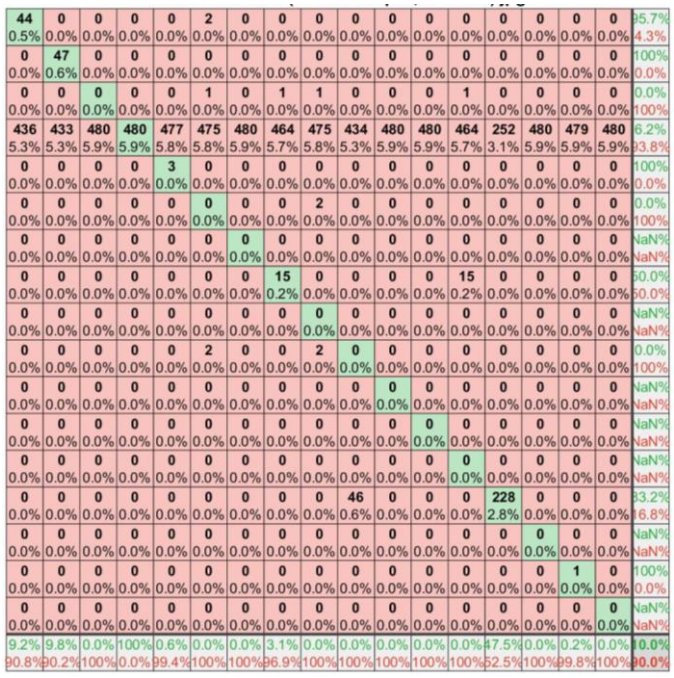

Figura 29.- Matriz de confusión de la salida de la validación de la red con dos capas ocultas y 17 fallos.

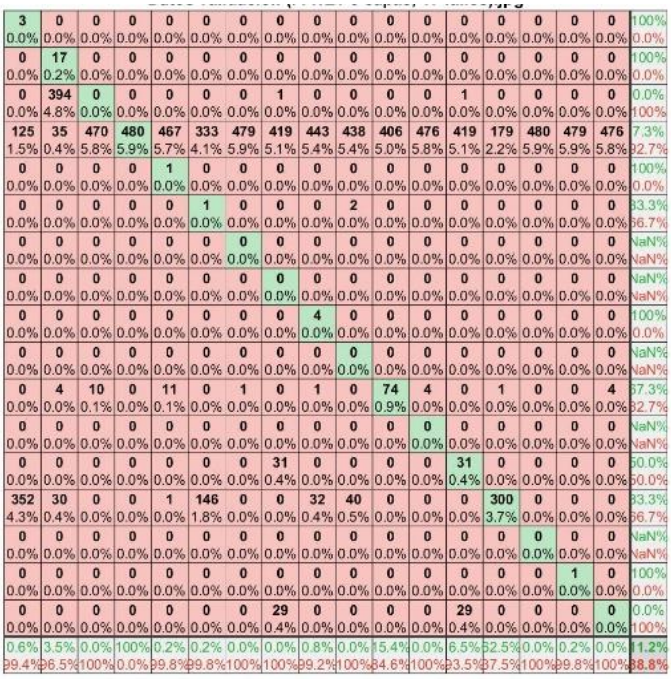

Figura 30.- Matriz de confusión de la salida de la validación de la red con tres capas ocultas y 17 fallos.

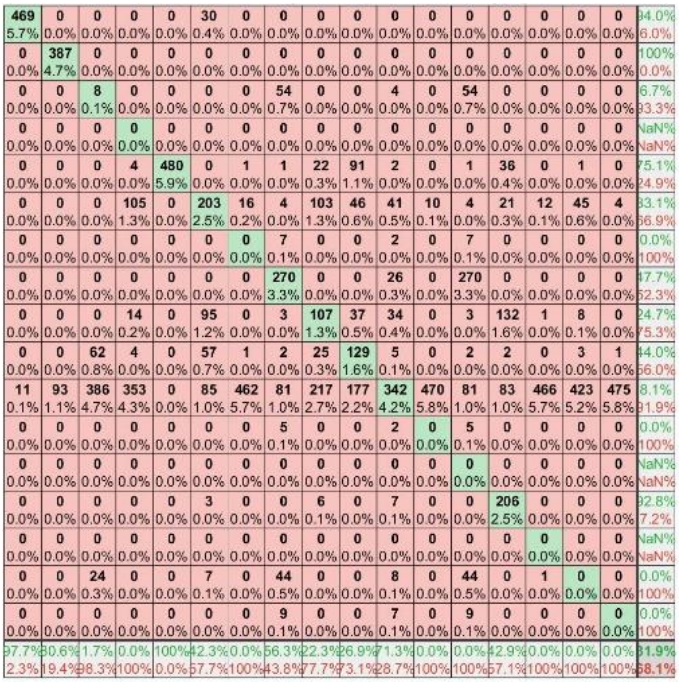

Figura 31.- Matriz de confusión de la salida de la validación de la red con cuatro capas ocultas y 17 fallos.

Las redes de propagación hacia delante no dan resultados aceptables, Como se puede ver, algunos fallos se clasifican correctamente, como el 1 y el 2, que tienen porcentajes de clasificación correcta altos (mayores del 95%) en

todos los caos, pero luego se tienen fallos que apenas tienen tasas de acierto en la clasificación del 10%, como, por ejemplo, el fallo 4, que se clasifica mal en todas las experiencias con 17 fallos. Además, los porcentajes de clasificación total son demasiado bajos para ser aceptables en un entorno industrial: en torno al 40% de los fallos no se identifican correctamente para 11 fallos, y para 17 fallos las tasas son aún menores. Todo esto, independientemente del número de capas ocultas con el que se trabaje, aunque sí que parece haber una correlación entre un mayor porcentaje de acierto y más capas ocultas. Un incremento en el número de neuronas estabiliza las salidas, pero solo hasta cierto punto; no se consigue que la clasificación sea correcta. En la tabla 5 y 6 se recogen los resultados de la experimentación. Se puede ver que no hay gran diferencia entre experiencias, siendo las redes de una y tres capas similares en cuanto a resultados, en ambos casos. Los resultados eran previsibles, ya que las redes de propagación hacia delante tienen poca capacidad de procesamiento de gran cantidad de datos para la clasificación de estos, como está explicado en el capítulo 2. Los mejores resultados se encuentran al utilizar 3 capas ocultas para 11 fallos, y 4 para 17.

Tabla 5.- Porcentaje de clasificación total de los datos de entrada usando redes de propagación hacia delante con 11 fallos.

Número de capas ocultas	Datos originales	Datos prueba	Datos validación
1	90.6	68.9	73
2	92.2	58.1	65
3	96.1	62.9	73
4	94	63.8	67.8

Tabla 6.- Porcentaje de clasificación total de los datos de entrada usando redes de propagación hacia delante con 17 fallos.

Número de capas ocultas	Datos originales	Datos prueba	Datos validación
1	71.9	20.5	8.8
2	54.6	35.6	10
3	70.4	15.7	11.2
4	60.9	33.1	31.9

3.2. Redes de aprendizaje profundo. Deep learning

Las redes de aprendizaje profundo se realizaron encadenando *autoencoders* como ya se ha explicado en el estudio teórico. Los *autoencoders* se construyeron con dos y tres capas ocultas, y se probó tanto con 11 fallos, como con 17. También se probó tanto a usar una capa final de clasificación del tipo *softmax* y a usar una red de propagación hacia delante con una capa oculta en su lugar. La arquitectura de las redes creadas se recoge en la tabla 7, indicando el número de neuronas de cada capa oculta, así como la capa de clasificación. Usando la capa *softmax*, sus neuronas coinciden con la de la capa de salida, y usando una red de propagación hacia delante, se utilizarán 25 neuronas en la capa oculta, y el número de neuronas en la capa de salida coincidirá con el número de fallos a clasificar.

Tabla 7.- Arquitectura de la red de deep learning. *Las dos primeras filas corresponden a la capa softmax; las dos últimas a la red* de propagación hacia delante.

	Capa oculta 1	Capa oculta 2	Capa oculta 3	Capa de clasificación	
Capa de entrada (52)	40	30	-	11 ó 17	Capa de salida (11 ó 17)
	40	30	15	11 ó 17	
	40	30	-	25	
	40	30	15	25	

El comienzo del procedimiento experimental es similar al seguido hasta ahora. Primero hay que crear la red, entrenarla y probarla con datos no vistos en el entrenamiento, y luego hay que validar la red con nuevos datos. Se usarán los ficheros de validación y los de entrenamiento divididos de la forma ya explicada, las observaciones 161 a la 750 se utilizarán para crear la herramienta, y las observaciones 751 a la 960 (210) serán usadas para probar su funcionamiento.

Tras normalizar los datos de entrada y crear la matriz de salida deseada, Y_d, de la misma manera que en situaciones anteriores (teniendo en cuenta que ahora las dimensiones de la matriz ya que se utilizan 11 ó 17 fallos), se procede a crear los *autoencoders* y a extraer sus capas ocultas para la concatenación. Para crear el *autoencoder*, hay que especificar la entrada deseada, generada como en las redes de propagación hacia delante, teniendo en cuenta el número de fallos que se quiere clasificar y el numero deseado de neuronas en la capa oculta. La salida deseada no hay que generarla en este tipo de red neuronal, ya que en este caso la salida es igual que la entrada. También se especifican como parámetros el número máximo de iteraciones, 600, para evitar

sobreajustes. Se usará el algoritmo de entrenamiento por defecto basado en gradiente conjugado, el mismo que se usó para las redes de propagación hacia delante multicapa, por su rapidez. Una vez entrenado el primer *autoencoder*, se extrae su capa oculta. Esta capa servirá de entrada al siguiente *autoencoder,* al que se le volverá a extraer la capa oculta si procede. Una vez se obtiene la capa oculta del último *autoencoder*, se crea la capa de clasificación.

La capa de clasificación se entrena de manera supervisada, por lo que se debe aportar tanto la matriz de entrada, en este caso la última capa oculta extraída, como la matriz de salida deseada, creada de la misma forma que casos anteriores. En el caso de usar una capa *softmax,* se entrena esta única capa con un algoritmo adecuado, y para la red de propagación hacia delante se sigue el mismo procedimiento que en casos anteriores: se crea la red de propagación hacia delante con un número determinado de neuronas en la capa oculta y se entrena con el algoritmo basado en el gradiente conjugado usando las entradas y salidas.

Para formar la red de aprendizaje profundo se concatenan todas las capas ocultas extraídas junto con la capa de clasificación usada, para formar la red neuronal de deep learning. Un ejemplo se puede ver en la figura 32: una red neuronal formada por dos capas ocultas. de 40 y 30 neuronas respectivamente, extraídas de los *autoencoders* correspondientes, unidas entre sí y junto a una capa *softmax*, que procesan 52 entradas diferentes, para procesarlas en 11 salidas diferentes, una por cada fallo que se desea diagnosticar.

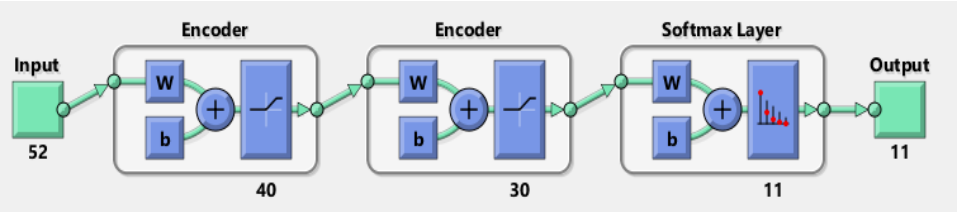

Figura 32.- Red neuronal de deep learning *con dos capas ocultas y una* softmax.

Ahora esta red neuronal formada por la concatenación de capas ocultas extraídas se puede usar para procesar los datos, pero como cada parte se ha pre-entrenado por su cuenta, no funciona de forma óptima. Hay que hacer un ajuste más preciso de los pesos y las *bias* de la red, con un entrenamiento global de la red. Antes de finalizar, se hace un entrenamiento usando como entradas deseadas los datos de todos los fallos que se quieran detectar, como datos de salida deseada la matriz Y_d formada por unos y ceros, ya comentada anteriormente, y se entrena con el algoritmo basado en el gradiente conjugado.

En este entrenamiento, se limita las iteraciones a 4000, para reducir la carga computacional y evitar sobreajustes. En el entrenamiento de las redes de propagación hacia delante, debido a que requieren más recursos de la CPU del ordenador, se habilitó a MATLAB™ con la capacidad de usar la GPU para ayudar en los cálculos y reducir el tiempo de entrenamiento.

Acabado el entrenamiento, se realiza una prueba de la red tanto con los datos usados para su creación y entrenamiento, como los reservados para la prueba. Luego, se valida la red en otro fichero con el procedimiento habitual, cargando los ficheros de fallos de validación y creando una matriz de salida deseada para los nuevos datos.

Para analizar los resultados, se comparan los casos con 11 fallos y 17 fallos, y dentro de cada uno se compara usar una capa de clasificación tipo softmax o de propagación hacia delante. Las figuras 33 y 34, muestran los resultados de validación de estas redes. Como se puede ver, los porcentajes de clasificación total son bastante mayores con 11 fallos (60.2%) que con 17 (14.9 %), aunque en ambos casos son demasiado bajos para tener un diagnóstico de fallos eficaz.

Los resultados son considerablemente mejores que las redes de propagación hacia delante en algunos casos, y peores en otros. Por ejemplo, usando una capa *softmax*, y con 11 fallos, la clasificación de los datos originales con los que se creó el primer *autoencoder* es prácticamente perfecta. Sin embargo, la clasificación para los datos de prueba y los de validación es bastante pobre (menor del 50% y del 65% respectivamente). En general, 17 fallos parecen ser demasiado para las redes creadas, siendo sus resultados peores que con 11 fallos (tablas 8 y 9). Es difícil escoger una combinación de parámetros con resultados aceptables, pero una red compuesta de dos capas ocultas con una capa de clasificación de propagación hacia delante diagnosticando 11 fallos parece ser la que mejores resultados proporciona, además de ser consistentes entre repetidas pruebas.

Tabla 8.- Porcentaje de clasificación total para una red de deep learning *con 11 fallos.*

Número de capas ocultas	Red de propagación hacia delante			Softmax		
	Datos originales	Datos prueba	Datos validación	Datos originales	Datos prueba	Datos validación
2	95.6	64.6	72.8	100	41.3	62.7
3	95.7	60.3	66.3	99.9	37.1	60.2

Tabla 9.- Porcentaje de clasificación total para una red de deep learning *con 17 fallos.*

Número de capas ocultas	Red de propagación hacia delante			Softmax		
	Datos originales	Datos prueba	Datos validación	Datos originales	Datos prueba	Datos validación
2	71.2	31.9	31.6	89.3	17.9	16.6
3	37.7	28.5	31.2	74.7	34.6	24.9

378 7.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**11** 0.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	97.2% 2.8%
0 0.0%	**464** 8.8%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**3** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	98.9% 1.1%
0 0.0%	**0** 0.0%	**136** 2.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**25** 0.5%	**0** 0.0%	**0** 0.0%	**0** 0.0%	84.5% 15.5%
0 0.0%	**0** 0.0%	**0** 0.0%	**419** 7.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	99.8% 0.2%
13 0.2%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**331** 6.3%	**6** 0.1%	**21** 0.4%	**4** 0.1%	**24** 0.5%	**4** 0.1%	**0** 0.0%	81.9% 18.1%
1 0.0%	**4** 0.1%	**0** 0.0%	**0** 0.0%	**14** 0.3%	**212** 4.0%	**28** 0.5%	**22** 0.4%	**18** 0.3%	**108** 2.0%	**4** 0.1%	51.6% 48.4%
2 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**2** 0.0%	**4** 0.1%	**43** 0.8%	**17** 0.3%	**1** 0.0%	**4** 0.1%	**0** 0.0%	58.1% 41.9%
0 0.0%	**0** 0.0%	**333** 6.3%	**0** 0.0%	**1** 0.0%	**2** 0.0%	**13** 0.2%	**219** 4.1%	**1** 0.0%	**0** 0.0%	**2** 0.0%	38.4% 61.6%
83 1.6%	**5** 0.1%	**0** 0.0%	**27** 0.5%	**76** 1.4%	**121** 2.3%	**192** 3.6%	**54** 1.0%	**317** 6.0%	**153** 2.9%	**18** 0.3%	30.3% 69.7%
2 0.0%	**6** 0.1%	**10** 0.2%	**32** 0.6%	**54** 1.0%	**120** 2.3%	**180** 3.4%	**75** 1.4%	**110** 2.1%	**208** 3.9%	**3** 0.1%	26.0% 74.0%
1 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**2** 0.0%	**1** 0.0%	**3** 0.1%	**64** 1.2%	**8** 0.2%	**3** 0.1%	**453** 8.6%	84.7% 15.3%
78.8% 21.3%	96.7% 3.3%	28.3% 71.7%	87.3% 12.7%	69.0% 31.0%	44.2% 55.8%	9.0% 91.0%	45.6% 54.4%	66.0% 34.0%	43.3% 56.7%	94.4% 5.6%	**60.2%** **39.8%**

Figura 33.- Matriz de confusión de la validación de una red de deep learning creada con tres autoencoders para detectar 11 fallos

478 5.9%	8 0.1%	172 2.1%	415 5.1%	338 4.1%	247 3.0%	216 2.6%	180 2.2%	202 2.5%	218 2.7%	168 2.1%	194 2.4%	180 2.2%	80 1.0%	187 2.3%	222 2.7%	228 2.8%	12.8% 87.2%
0 0.0%	9 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0.0% 100%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0.0% 100%
0 0.0%	1 0.0%	0 0.0%	0 0.0%	142 1.7%	1 0.0%	0 0.0%	0 0.0%	2 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	24 0.3%	0 0.0%	0 0.0%	0 0.0%	33.0% 17.0%
2 0.0%	42 0.5%	306 3.8%	65 0.8%	0 0.0%	219 2.7%	261 3.2%	135 1.7%	222 2.7%	209 2.6%	306 3.8%	286 3.5%	135 1.7%	185 2.3%	291 3.6%	256 3.1%	251 3.1%	6.9% 93.1%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	142 1.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	142 1.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	50.0% 50.0%
0 0.0%	8 0.1%	2 0.0%	0 0.0%	0 0.0%	13 0.2%	3 0.0%	2 0.0%	53 0.6%	42 0.5%	4 0.0%	0 0.0%	2 0.0%	11 0.1%	2 0.0%	1 0.0%	1 0.0%	86.8% 63.2%
0 0.0%	412 5.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.1%	0 0.0%	0 0.0%	0 0.0%	14 0.2%	0 0.0%	0 0.0%	0 0.0%	1.2% 98.8%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	18 0.2%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	18 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5.3% 94.7%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	50.0% 50.0%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.1%	0 0.0%	0 0.0%	0 0.0%	166 2.0%	0 0.0%	0 0.0%	0 0.0%	97.1% 2.9%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0.0% 100%
99.6% 0.4%	1.9% 98.1%	0.0% 100%	0.0% 100%	29.6% 70.4%	45.6% 54.4%	0.0% 100%	29.6% 70.4%	11.0% 89.0%	1.0% 99.0%	0.4% 99.6%	0.0% 100%	0.2% 99.8%	34.6% 65.4%	0.0% 100%	0.0% 100%	0.0% 100%	14.9% 85.1%

Figura 34.- Matriz de confusión para la validación de una red de aprendizaje profundo creada con 3 autoencoders para detectar 17 fallos

3.3. Bosque aleatorio.

Para acabar la experimentación, se utilizaron los árboles de decisiones agregados, o bosques aleatorios. Se probó con diferente número de árboles para encontrar un equilibrio entre el sobreajuste producido al utilizar un número alto y la clasificación pobre que se da al utilizar pocos. Se probó tanto con 11 fallos, como con 17. Es necesario primero crear y entrenar el bosque aleatorio con datos de entrenamiento, luego validar el bosque obtenido con datos distintos del entrenamiento y finalmente probar el resultado con datos distintos cogidos de la planta.

La creación del bosque es sencilla. Se cogen los datos de fallos que se quieren clasificar, y se dividen en sendas matrices para crear el bosque (X) y para probarlo (X_t), y se crea la matriz de salida deseada, y con estos datos se crea el bosque aleatorio. Tras la creación, se procede a probar el bosque. Después se procede a la validación de la herramienta recién creada. Los resultados se visualizan en una matriz de confusión. En la figura 35 se puede ver la forma de uno de los cientos de árboles que componen al bosque aleatorio.

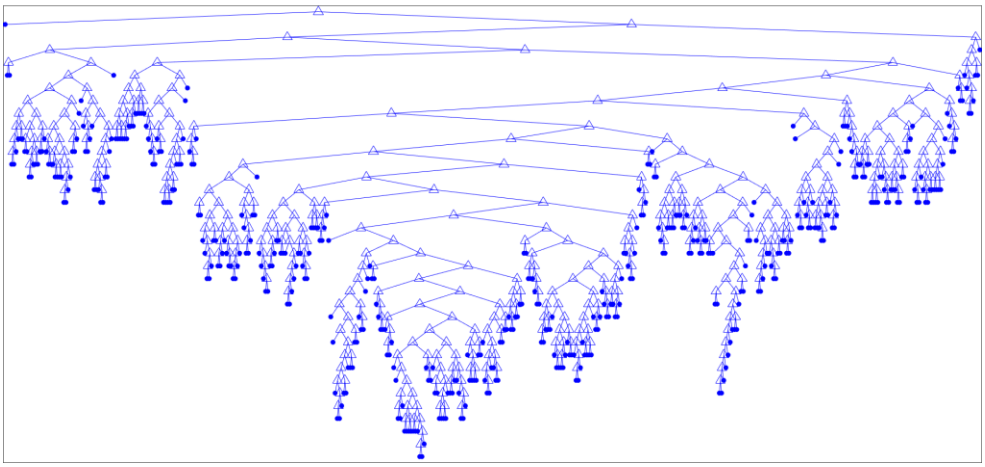

Figura 35.- Estructura de un árbol ensamblado en el bosque

Para analizar los resultados se comparan para los casos de 11 y 17 fallos. Un aumento en el número de fallos a diagnosticar implica una peor clasificación, como se puede ver en las tablas 10 y 11. A destacar que los datos originales son clasificados perfectamente en todos los casos, no como con las redes neuronales. Con 11 fallos y 200 árboles se consiguen los mejores resultados. Se puede ver como para números de árboles altos los porcentajes de clasificaciones correctas no aumenta significativamente. En las figuras 36 y 37 se puede ver la matriz de confusión para los datos de validación del mejor bosque aleatorio para 11 y 17 fallos respectivamente. Como se puede ver, con 11 fallos el porcentaje de clasificación total es mayor que con 17. A pesar de ello, es importante ver que los porcentajes de verdaderos positivos es alta para ambos casos, y para la mayoría de los fallos, lo cual contrasta con las técnicas previamente utilizadas. Ahora todos los fallos tienen porcentajes de verdaderos positivos altos, y no hay ninguno que presente falsos positivos en tanta cantidad como anteriormente.

Tabla 10.- Porcentaje de clasificación total para el bosque aleatorio con 11 fallos

Numero de arboles	Datos originales	Datos prueba	Datos validación
50	100	83.5	82
100	100	84.8	82.7
150	100	84.6	82.8
200	100	85	83

250	100	84.8	83.1
300	100	84.5	83.2
500	100	84.8	83.2

Tabla 11.- Porcentaje de clasificación total para el bosque aleatorio con 17 fallos

Numero de arboles	Datos originales	Datos prueba	Datos validación
50	100	73.1	77.4
100	100	73.7	77.7
150	100	73.8	77.6
200	100	73.6	77.4
250	100	74.3	77.5
300	100	74.1	77.7
500	100	74.6	77.7

474 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**36** 0.7%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	92.9% 7.1%
0 0.0%	**466** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
0 0.0%	**0** 0.0%	**473** 9.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**46** 0.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	91.1% 8.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**437** 8.3%	**0** 0.0%	**11** 0.2%	**11** 0.2%	**2** 0.0%	**16** 0.3%	**10** 0.2%	**0** 0.0%	89.7% 10.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**478** 9.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**3** 0.1%	**0** 0.0%	**0** 0.0%	99.4% 0.6%
0 0.0%	**2** 0.0%	**0** 0.0%	**8** 0.2%	**0** 0.0%	**280** 5.3%	**0** 0.0%	**0** 0.0%	**50** 0.9%	**195** 3.7%	**0** 0.0%	52.3% 47.7%
6 0.1%	**12** 0.2%	**2** 0.0%	**8** 0.2%	**0** 0.0%	**18** 0.3%	**449** 8.5%	**35** 0.7%	**20** 0.4%	**31** 0.6%	**0** 0.0%	77.3% 22.7%
0 0.0%	**0** 0.0%	**4** 0.1%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**10** 0.2%	**383** 7.3%	**10** 0.2%	**7** 0.1%	**5** 0.1%	91.2% 8.8%
0 0.0%	**0** 0.0%	**0** 0.0%	**27** 0.5%	**2** 0.0%	**79** 1.5%	**8** 0.2%	**0** 0.0%	**367** 7.0%	**138** 2.6%	**0** 0.0%	59.1% 40.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**55** 1.0%	**2** 0.0%	**0** 0.0%	**14** 0.3%	**99** 1.9%	**0** 0.0%	58.2% 41.8%
0 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**14** 0.3%	**0** 0.0%	**0** 0.0%	**475** 9.0%	96.9% 3.1%
98.8% 1.2%	97.1% 2.9%	98.5% 1.5%	91.0% 9.0%	99.6% 0.4%	58.3% 41.7%	93.5% 6.5%	79.8% 20.2%	76.5% 23.5%	20.6% 79.4%	99.0% 1.0%	**83.0%** **17.0%**

Figura 36.- Matriz de confusión de datos de validación para un bosque de 500 árboles y 17 fallos.

473 5.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**36** 0.4%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	92.9% 7.1%
0 0.0%	**463** 5.7%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	99.6% 0.4%
0 0.0%	**0** 0.0%	**476** 5.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**45** 0.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	91.4% 8.6%
0 0.0%	**0** 0.0%	**0** 0.0%	**402** 4.9%	**0** 0.0%	**19** 0.2%	**11** 0.1%	**4** 0.0%	**17** 0.2%	**11** 0.1%	**0** 0.0%	**7** 0.1%	**1** 0.0%	**4** 0.0%	**1** 0.0%	**7** 0.1%	**0** 0.0%	83.1% 16.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**478** 5.9%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**3** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	99.2% 0.8%
0 0.0%	**5** 0.1%	**0** 0.0%	**10** 0.1%	**0** 0.0%	**264** 3.2%	**0** 0.0%	**0** 0.0%	**45** 0.6%	**164** 2.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	53.9% 46.1%
0 0.0%	**3** 0.0%	**0** 0.0%	**10** 0.1%	**0** 0.0%	**3** 0.0%	**200** 2.5%	**4** 0.0%	**4** 0.0%	**30** 0.4%	**0** 0.0%	**46** 0.6%	**2** 0.0%	**4** 0.0%	**0** 0.0%	**33** 0.4%	**3** 0.0%	58.5% 41.5%
0 0.0%	**3** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**354** 4.3%	**0** 0.0%	**2** 0.0%	**9** 0.1%	**3** 0.0%	**5** 0.1%	**3** 0.0%	**6** 0.1%	**0** 0.0%	**1** 0.0%	91.0% 9.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**27** 0.3%	**2** 0.0%	**61** 0.7%	**4** 0.0%	**0** 0.0%	**346** 4.2%	**142** 1.7%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**35** 0.4%	**0** 0.0%	**0** 0.0%	**0** 0.0%	56.1% 43.9%
0 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**55** 0.7%	**1** 0.0%	**0** 0.0%	**17** 0.2%	**111** 1.4%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**10** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	57.2% 42.8%
0 0.0%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**16** 0.2%	**0** 0.0%	**0** 0.0%	**452** 5.5%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	96.0% 4.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**7** 0.1%	**0** 0.0%	**1** 0.0%	**205** 2.5%	**7** 0.1%	**9** 0.1%	**2** 0.0%	**0** 0.0%	**325** 4.0%	**3** 0.0%	**3** 0.0%	**3** 0.0%	**30** 0.4%	**22** 0.3%	52.7% 47.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**4** 0.0%	**3** 0.0%	**0** 0.0%	**18** 0.2%	**1** 0.0%	**431** 5.3%	**0** 0.0%	**4** 0.0%	**8** 0.1%	**0** 0.0%	91.7% 8.3%
0 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**17** 0.2%	**0** 0.0%	**1** 0.0%	**12** 0.1%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**343** 4.2%	**0** 0.0%	**0** 0.0%	**0** 0.0%	91.5% 8.5%
4 0.0%	**6** 0.1%	**0** 0.0%	**16** 0.2%	**0** 0.0%	**12** 0.1%	**29** 0.4%	**41** 0.5%	**16** 0.2%	**17** 0.2%	**1** 0.0%	**71** 0.9%	**22** 0.3%	**37** 0.5%	**454** 5.6%	**71** 0.9%	**16** 0.2%	65.8% 44.2%
3 0.0%	**0** 0.0%	**0** 0.0%	**5** 0.1%	**0** 0.0%	**11** 0.1%	**22** 0.3%	**4** 0.0%	**8** 0.1%	**0** 0.0%	**0** 0.0%	**19** 0.2%	**15** 0.2%	**36** 0.4%	**11** 0.1%	**330** 4.0%	**1** 0.0%	71.0% 29.0%
0 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**6** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**8** 0.1%	**0** 0.0%	**2** 0.0%	**1** 0.0%	**1** 0.0%	**437** 5.4%	95.8% 4.2%
98.5% 1.5%	96.5% 3.5%	99.2% 0.8%	83.8% 16.2%	99.6% 0.4%	55.0% 45.0%	41.7% 58.3%	73.8% 26.2%	72.1% 27.9%	23.1% 76.9%	94.2% 5.8%	67.7% 32.3%	89.8% 10.2%	71.5% 28.5%	94.6% 5.4%	68.8% 31.3%	91.0% 9.0%	**77.7%** **22.3%**

Figura 37.- Matriz de confusión de los datos de validación para un bosque con 200 árboles y 11 fallos.

3.4. Comparación de resultados.

En este apartado, se contrastarán los mejores resultados obtenidos de cada técnica para establecer que técnica de aprendizaje supervisado es la que mejor diagnostica los fallos de la planta. Para ello, se usarán los resultados con mayores porcentajes de clasificación correcta total de las matrices de confusión de las experiencias. La comparación está en la tabla 12 y 13.

Tabla 12.- Comparación de resultados para la clasificación de 11 fallos

Técnica	Datos originales	Datos prueba	Datos validación
Red propagación hacia delante con 3 capas ocultas	96.1	62.9	73
Red de aprendizaje profundo con 2 capas ocultas y una capa de clasificación de propagación hacia delante	95.6	64.6	72.8
Bosque aleatorio con 200 árboles	100	85	83

Tabla 13.- Comparación de resultados para la clasificación de 17 fallos

Técnica	Datos originales	Datos prueba	Datos validación
Red propagación hacia delante con 4 capas ocultas	60.9	33.1	31.9
Red de aprendizaje profundo con 2 capas ocultas y una capa de clasificación de propagación hacia delante	71.2	31.9	31.6
Bosque aleatorio con 500 árboles	100	74.6	77.7

Se puede ver que el bosque aleatorio ha dado los mejores resultados en cuanto a clasificación de fallo, tanto para 11 como para 17 fallos, teniendo los porcentajes de clasificación total correcta más altos de todas las técnicas probadas. Cuantos más datos tiene que procesar, más árboles se necesitan, de ahí que el bosque utilizado para los 17 fallos tenga 500 árboles. Las redes neuronales multicapa y las redes de aprendizaje profundo no han dado buenos resultados. El resultado de ambas arquitecturas es prácticamente el mismo, por lo que preentrenar los *autoencders* no ha redundado en un beneficio del sistema. Por otro lado, quizás este mal resultado pueda ser debido a la cantidad de datos de entrenamiento de fallos que se tiene, ya que se tienen muchos fallos que se quieren clasificar y pocas observaciones de cada fallo. Una posible mejora sería conseguir más datos de entrenamiento de fallos para probar estas redes y ver si se obtiene un mejor resultado.

CAPITULO V: CONCLUSIONES. TRABAJO FUTURO

1. Conclusiones

Debido a la nueva industria 4.0 la cantidad de datos con los que se trabaja hoy en día requiere de técnicas sofisticadas y potentes en cuestión de procesamiento de información. En este trabajo se han utilizado los métodos FDD, en el contexto del control de calidad, siendo el principal objetivo experimentar con técnicas estadísticas como es el PCA para detectar fallos y varias técnicas de aprendizaje automático para su posterior clasificación y diagnóstico. La experimentación se ha basado en unos datos provenientes de un modelo estándar de proceso, la planta Tennessee Eastman.

Se ha creado una aplicación PCA fuera de línea con datos de comportamiento normal para definir unos umbrales estadísticos, tras lo cual se ha realizado la aplicación online para procesar los datos de fallo, y comprobar que los umbrales cumplían su función. También se han analizado los vectores de residuos del problema para realizar un diagrama de contribuciones con los que analizar que variables influían en que fallos. Los resultados obtenidos han sido correctos, detectando el modelo creado rápidamente todos los fallos con los que se ha probado. Además, con el diagrama de contribuciones se ha podido relacionar cada fallo con las variables que más influían en él, como por ejemplo el fallo 1, la ratio A/C de alimentación, que se ve más afectada por la presión de la columna stripper y el compresor.

Se han desarrollado varias técnicas de aprendizaje automático para diagnosticar los fallos una vez detectados, mediante su clasificación en clases. Se empezó con redes de propagación hacia delante con una sola capa oculta, y tras ver que solo soportaban clasificar cuatro fallos antes de dar errores importantes, se pasó a redes de propagación hacia delante con varias capas ocultas para intentar clasificar 11 fallos. También se han creado redes de aprendizaje profundo mediante la concatenación de *autoencoders*, probando a crearlas con una capa de clasificación *softmax* o con una red de propagación hacia delante, y se han probado con 11 y 17 fallos. Por último, se han creado bosques aleatorios ensamblando árboles de decisión para clasificar 11 y 17 fallos.

Las redes de propagación hacia delante no han dado buenos resultados en ningún caso, y las redes de aprendizaje profundo han funcionado mejor con tan solo dos capas ocultas con una capa de clasificación de propagación hacia delante para clasificar 11 que 17 fallos. Los mejores resultados han resultado de usar los bosques aleatorios, tanto para 11 como para 17 fallos, indicando la potencia de estas técnicas en la clasificación binaria. También se puede considerar que la cantidad de datos que se tiene para entrenar redes de aprendizaje profundo no son suficientes y esta puede ser la razón por la que se obtienen resultados desfavorables en todas las técnicas. Además, alguno de

los fallos es demasiado parecidos entre sí, con variaciones similares, lo que dificulta su clasificación en la clase correcta.

2. Trabajo futuro

Respecto al trabajo futuro, se deberían explorar más los bosques aleatorios como técnicas de clasificación de fallos, dado que han tenido los mejores resultados. Aumentar el número de árboles para superar los 500 utilizados en este trabajo o usar diferentes parámetros en su programación para restringir o dar más margen a la poda de los árboles, serían opciones sencillas con prometedores resultados. Las redes de aprendizaje profundo se podrían estudiar con más tiempo de computación y diferentes parámetros para definir las funciones de coste de los *autoencoders* concatenados, o se podrían estudiar otras arquitecturas de redes de aprendizaje profundo como la red convolucional, la red recurrente, etc. Además, se podría explorar la posibilidad de usar varios métodos de aprendizaje automático que se complementen para diagnosticar los fallos, además se podrían obtener más datos de cada tipo de fallo, y entrenar las redes con mayor cantidad de datos (*big data*) para ver su resultado.

BIBLIOGRAFÍA

[1] V. Roblek, M. Meško, and A. Krapež, "A Complex View of Industry 4.0," *SAGE Open*, vol. 6, no. 2, 2016, doi: 10.1177/2158244016653987.

[2] M. S. Reis and G. Gins, "Industrial process monitoring in the big data/industry 4.0 era: From detection, to diagnosis, to prognosis," *Processes*, vol. 5, no. 3, 2017, doi: 10.3390/pr5030035.

[3] M. J. De la Fuente Aparicio, "Control estadístico de procesos.," *Apuntes de la asignatura de Ingeniería Informática*. Universidad de Valladolid. Ingeniería de Sistemas y Automática., 2019.

[4] S. J. Qin, "Data-driven Fault Detection and Diagnosis for Complex Industrial Processes," *IFAC Proc. Vol.*, vol. 42, no. 8, pp. 1115–1125, 2009, doi: https://doi.org/10.3182/20090630-4-ES-2003.00184.

[5] R. Rivas-Perez (1), P. Zubieta (2), and H. Garcini (1), "Review. Detección y diagnóstico automático de fallos en procesos industriales." (1) Departamento de Automática y Computación Facultad de Ingeniería Eléctrica Instituto Superior Politécnico José Antonio Echeverría; (2) Facultad de Ingeniería, Universidad Técnica de Oruro, 2015.

[6] W. A. Shewhart, *Economic control of quality of manufactured product*. Milwaukee, Wis. ASQ Quality Press, 1931.

[7] Z. G. Stoumbos, M. R. Reynolds, and W. H. Woodall, "Ch. 13. Control chart schemes for monitoring the mean and variance of processes subject to sustained shifts and drifts," in *Statistics in Industry*, vol. 22, Elsevier, 2003, pp. 553–571.

[8] D. Penfield, "Gráfico de control genérico," 2019. wikipedia.org (accessed Apr. 15, 2020).

[9] L. I. Smith, *A tutorial on Principal Components Analysis Introduction*. 2002.

[10] L. Chiang, E. Russell, and R. Braatz, "Fault Detection and Diagnosis in Industrial Systems," *Meas. Sci. Technol. - MEAS SCI TECHNOL*, vol. 12, 2001, doi: 10.1088/0957-0233/12/10/706.

[11] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997.

[12] T. Villegas Berbesi, "Tesis doctoral. Aplicación de técnicas robustas para detección y diagnóstico de fallos." Universidad de Valladolid. Ingeniería de Sistemas y Automática., 2012.

[13] S. Cash and R. Yuste, "Linear Summation of Excitatory Inputs by CA1 Pyramidal Neurons," *Neuron*, vol. 22, no. 2, pp. 383–394, Feb. 1999, doi: 10.1016/S0896-6273(00)81098-3.

[14] A. Cartas, "Diagrama de un perceptrón con cinco señales de entrada.,"

2015. https://es.wikipedia.org/wiki/Perceptrón (accessed Jun. 10, 2020).

[15] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control. Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989, doi: 10.1007/BF02551274.

[16] S. Feng, H. Zhou, and H. Dong, "Using deep neural network with small dataset to predict material defects," *Mater. Des.*, vol. 162, pp. 300–310, 2019, doi: 10.1016/j.matdes.2018.11.060.

[17] N. G. Lo, J. M. Flaus, and O. Adrot, "Review of Machine Learning Approaches in Fault Diagnosis applied to IoT Systems," *2019 Int. Conf. Control. Autom. Diagnosis, ICCAD 2019 - Proc.*, 2019, doi: 10.1109/ICCAD46983.2019.9037949.

[18] C. Aldrich and L. Auret, *Fault detection and diagnosis with random forest feature extraction and variable importance methods*, vol. 43, no. 9 PART 1. IFAC, 2010.

[19] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *J. Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012, doi: 10.1016/j.jprocont.2012.06.009.

[20] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993, doi: 10.1016/0098-1354(93)80018-I.

[21] S. Krishnannair and C. Aldrich, "Fault detection in the Tennessee Eastman benchmark process with nonlinear singular spectrum analysis," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8005–8010, 2017, doi: https://doi.org/10.1016/j.ifacol.2017.08.1223.

[22] J. I. Medrano Sanz, "Trabajo de fin de grado. Estudio de técnicas de clasificación para la detección y diagnóstico de fallos ." Universidad de Valladolid, 2019.

[23] R. Braatz, "Tennessee Eastman process (TEP) training and testing data files," *Massachusetts Institute of Technology Advanced Manufacturing Systems*, 2020. http://web.mit.edu/braatzgroup/links.html.

[24] "Arbol de decision simple," 1997. http://informatica.uv.es/iiguia/AED/oldwww/EDS/9697/PRACT_07/pr97_07.html (accessed Jul. 10, 2020).