Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Monitorización distribuida de un proceso para mejora de su calidad mediante técnicas de inteligencia computacional/Deep learning: RAE

Autor:

Fernández Fernández, Pablo

Tutor:

De la Fuente, Aparicio, María Jesús

Departamento de Ingeniería de Sistemas y Automática

Valladolid, Diciembre 2025.

Contenido

..... 1

Capítulo 1. Introducción y objetivos..... 7

1.1. Introducción 7

1.2. Objetivos 7

1.3. Organización de la memoria 8

Capítulo 2. Estudio teórico 9

2.1 Control de Calidad..... 9

2.2 Control estadístico de procesos (SPC)..... 12

2.2.1 Variabilidad en el proceso de producción 12

2.2.2 Gráficas de Control 14

2.3 Control Estadístico de Procesos Multivariable (MSPC) 15

2.3.1 $T2$ de Hotelling..... 16

2.3.2 Error Q (Squared Prediction Error) 17

2.3.3 Uso combinado de $T2$ y Q..... 17

2.4 Análisis de componentes principales (PCA) 19

2.5 Redes Neuronales Artificiales (ANN)..... 23

2.5.1 Neurona artificial 24

2.5.2 Red de neuronas..... 25

2.5.3 Funciones de activación 25

2.5.4 Redes Neuronales Recurrentes (RNN) 28

2.6 Aprendizaje automático 30

2.6.1 Propagación hacia adelante (Forward propagation) 30

2.6.2 Cálculo del error 30

2.6.3 Retropropagación del error..... 31

2.6.4 Actualización de parámetros (Gradient Descent) 31

2.7 Autoencoders..... 33

2.7.1 Autoencoder Denso (Fully Connected Autoencoder)..... 36

2.7.2 Autoencoder Recurrente (RAE) variante LSTM (Long Short-Term Memory Autoencoder)..... 38

2.8 Monitorización distribuida..... 42

2.8.1 Subdivisión de variables en bloques mediante el método mínima redundancia máxima relevancia 43

2.8.2 Inferencia Bayesiana.....45
Capítulo 3. Proceso Tennessee-Eastman.....47
3.1. Origen y contexto histórico.....47
3.2. Descripción general del proceso.....47
3.3. Formulación matemática del modelo.....49
3.4. Condiciones de operación y control.....50
3.5. Aplicaciones y relevancia.....52
3.6. Limitaciones del modelo.....52
Capítulo 4. Aplicaciones y evaluación de métodos.....53
4.1 Introducción.....53
4.2 PCA.....54
4.2.1 Entrenamiento.....55
4.2.2 Detección de Fallos.....56
4.2.3 Resultados PCA.....58
4.3 Autoencoder.....59
4.3.1 Entrenamiento.....59
4.3.2 Detección de fallos.....62
4.3.3 Resultados.....64
4.4 Autoencoder Recurrente (RAE).....66
4.4.1 Entrenamiento.....66
4.4.2 Detección de Fallos.....69
4.4.3 Resultados.....71
4.5 RAE distribuido.....73
4.5.1 Entrenamiento.....73
4.5.2 detección de Fallos.....75
4.5.3 Resultados.....87
Capítulo 5. Conclusiones y trabajo futuro.....91
5.1 Conclusiones.....91
5.2 Trabajo futuro.....92
Bibliografía.....94

Resumen

Este estudio examina diversas metodologías de detección de errores fundamentadas en datos, que se utilizan para controlar la calidad en ámbitos industriales. Estos métodos son eficaces para extraer información relevante y mejorar la calidad de los procesos, debido a la enorme cantidad de datos producidos en la industria moderna.

El estudio se inicia con el análisis de los componentes principales, un método lineal para disminuir características que posibilita la creación de nuevas variables que pueden agrupar la mayor parte de la información del sistema y disminuir el número de dimensiones. Se pueden detectar fallos potenciales al comparar la conducta normal del proceso con datos atípicos, utilizando estadísticos multivariantes y la distribución de estos datos reducidos.

Además, se llevaron a cabo métodos de aprendizaje profundo y automático dirigidos a disminuir la dimensionalidad. En esta línea, los autoencoders posibilitan un aprendizaje sin supervisión de la información de la planta al captar relaciones no lineales entre las variables y dar paso a la identificación de anomalías. Los autoencoders recurrentes son una mejora de los autoencoders convencionales, ya que tienen información de los estados pasados del sistema y por lo tanto permiten una detección de errores más exacta y fiable.

Por último, dada la gran dimensionalidad de los datos que se recogen en las industrias actuales se creó una estrategia de detección de fallos distribuida para mejorar los modelos de detección de anomalías. Esta metodología divide las variables de la planta en diferentes bloques de manera automática, identifica anomalías en cada uno de ellos por separado y, a través de inferencia bayesiana, sintetiza la información adquirida.

Finalmente comentar, que los datos del proceso que se han usado para probar todas las técnicas desarrolladas has sido los datos del proceso Tennessee-Eastman, que se usa mucho en la literatura científica sobre detección de fallos.

Abstract

This study examines various data-driven error detection methodologies used for quality control in industrial settings. These methods are effective for extracting relevant information and improving process quality, owing to the enormous amount of data produced in modern industry. The data from the Tennessee-Eastman process, which is widely used in the scientific literature on fault detection, were employed for training and evaluating the models.

The study begins with principal component analysis, a linear method for reducing features that enables the creation of new variables capable of capturing most of the system's information and reducing the number of dimensions. Potential faults can be detected by comparing the normal behavior of the process with atypical data, using multivariate statistics and the distribution of these reduced data.

Additionally, deep and machine learning methods aimed at reducing dimensionality were employed. In this vein, autoencoders enable unsupervised learning of plant information by capturing nonlinear relationships between variables and paving the way for anomaly detection. Recurrent autoencoders are an improvement over conventional autoencoders, as they have information about the past states of the system so therefore, they enable more robust training and more accurate and reliable error detection.

At last, given the large dimensionality of the collected data, a distributed fault detection strategy was developed. This methodology automatically divides the plant's variables into different blocks, identifies anomalies in each block separately and through Bayesian inference synthesizes the acquired information.

Finally, it should be noted that the process data used to test all the developed techniques were from the Tennessee-Eastman process, which is widely used in the scientific literature on fault detection.

Capítulo 1. Introducción y objetivos

1.1. Introducción

La evolución de la industria, desde los talleres artesanales hasta las modernas plantas altamente automatizadas, ha estado siempre ligada a la necesidad de garantizar la calidad de los productos. A medida que aumentaron tanto el volumen como la complejidad de la producción, la inspección manual perdió eficacia, lo que llevó al desarrollo del control de calidad como disciplina propia. Su propósito es reducir la variabilidad de los procesos y prevenir desviaciones antes de que se traduzcan en fallos en el producto final.

En la actualidad, la disponibilidad masiva de datos industriales gracias a sensores y sistemas de monitorización en tiempo real, junto con la capacidad de cálculo existente, ha impulsado la aplicación de técnicas de inteligencia artificial. Estas permiten analizar simultáneamente cientos de variables, identificar patrones imposibles de detectar a simple vista y automatizar la detección de anomalías. De este modo, los sistemas no solo sustituyen la observación humana, sino que además aportan rapidez, objetividad y escalabilidad.

Por lo tanto, asegurar la calidad y la seguridad en la producción son dos objetivos de cualquier industria, que se consiguen mediante la implementación de métodos de detección y diagnóstico de fallos que detecten cualquier anomalía que aparezca en el funcionamiento de la planta. En este trabajo, se intenta usar tecnologías basadas en datos, y en concreto en la Inteligencia Artificial para aumentar el rendimiento de la planta, buscando aplicar estas herramientas al control de calidad de plantas industriales, y en concreto a la monitorización de dichos procesos.

1.2. Objetivos

El propósito central de este trabajo es desarrollar y comparar metodologías de detecciones de fallos basadas en datos, aplicables al control de calidad en plantas industriales. Para ello se contemplan dos líneas principales:

- El uso de técnicas clásicas de control estadístico de procesos, en particular el Análisis de Componentes Principales (PCA), que permite reducir la dimensionalidad de manera lineal y generar estadísticos de control multivariantes.

- La aplicación de enfoques de aprendizaje profundo, donde se exploran autoencoders recurrentes (RAE) como alternativa a los autoencoders convencionales. Estos modelos son capaces de aprender dependencias temporales y no lineales entre variables, lo que constituye una ventaja en entornos donde las dinámicas del proceso son relevantes.

Adicionalmente, se plantea una metodología distribuida que divide las variables en bloques, analiza cada uno por separado y posteriormente integra los resultados mediante inferencia bayesiana. En este trabajo, esta estrategia también se aplicará a los RAE, dando lugar a un apartado específico de RAE distribuido.

El estudio se valida utilizando el proceso Tennessee–Eastman, ampliamente reconocido en la literatura como banco de pruebas de algoritmos de detección de fallos. Esto permitirá evaluar el rendimiento comparativo de los métodos bajo condiciones equivalentes, considerando su precisión, fiabilidad y capacidad de generalización.

1.3. Organización de la memoria

La memoria está estructurada en cinco capítulos:

- En este Capítulo 1 se exponen la motivación, los objetivos y la organización del trabajo.
- El Capítulo 2 presenta los fundamentos teóricos necesarios, incluyendo control estadístico de procesos, detección de anomalías, estadísticos multivariantes, reducción dimensional y una introducción a redes neuronales y autoencoders.
- El Capítulo 3 describe el proceso Tennessee–Eastman y los conjuntos de datos empleados.
- En el Capítulo 4 se detallan las aplicaciones y la evaluación de las metodologías propuestas: PCA, Autoencoders, RAE y RAE distribuido.
- Finalmente, el Capítulo 5 recoge las principales conclusiones alcanzadas y plantea líneas de trabajo futuro.

Capítulo 2. Estudio teórico

2.1 Control de Calidad

Orígenes y primeras manifestaciones

La inquietud por la calidad de los productos y servicios tiene orígenes remotos: desde el control en las profesiones medievales hasta los códigos de leyes que requerían estándares mínimos. Estas primeras formas tenían como objetivo proteger la reputación del creador y al consumidor a través de marcas de conformidad e inspecciones [1], [2].

Con la Revolución Industrial, se incrementó la producción y surgieron nuevos desafíos: la variabilidad, las grandes series y el requerimiento de prevenir fallos. La inspección final fue la solución inicial, pero no bastó ante el tamaño y los costos de producción [2].

Invención del Control Estadístico

Walter A. Shewhart, un ingeniero de los laboratorios Bell en los años 20, es reconocido como el fundador del control estadístico de procesos (SPC), el cual se menciona más adelante, dado que un apartado de este trabajo se centra en este método para la detección de fallos.

Su propuesta consistió en emplear datos de producción para distinguir entre causas especiales (variaciones atribuibles a fallos específicos, como problemas técnicos o errores de los humanos) y causas comunes (la variabilidad natural que forma parte del proceso). Con este fin, incorporó las gráficas de control, que hacían posible observar de forma fácil si un proceso se mantenía estable o necesitaba intervención [3].

El concepto de mejora continua fue un componente fundamental en su planteamiento, que luego se concretó en el ciclo PDCA (Planificar-Hacer-Verificar-Actuar). Como se muestra en la Figura 1. Este ciclo sugiere planificar una acción, ponerla en práctica, comprobar sus resultados y proceder corrigiendo desviaciones. Deming, a pesar de que fue Shewhart quien introdujo la idea, es el responsable de popularizarla como herramienta de gestión universal [3], [4].

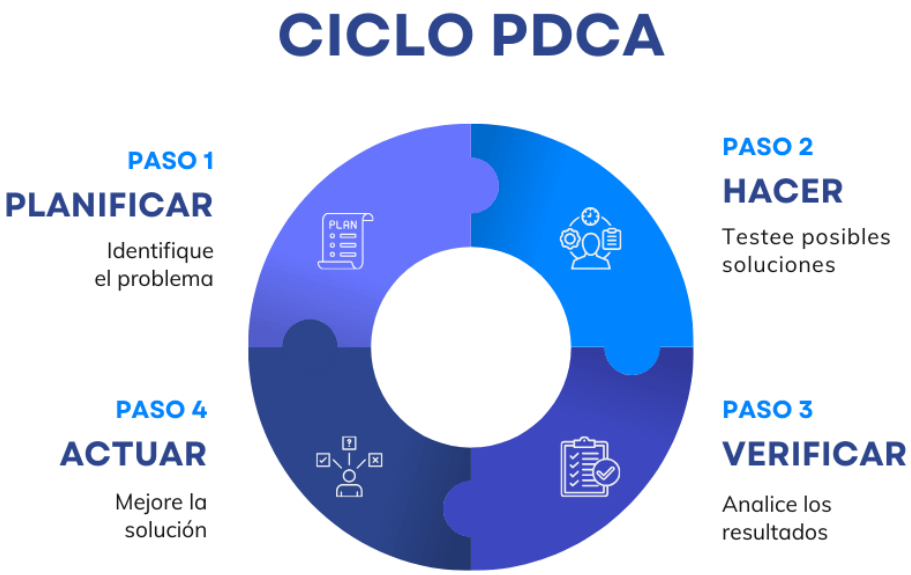


Figura 1. Ciclo PDCA de Shewhart [5]

Difusión global

En la administración de calidad, las contribuciones de Joseph M. Juran y W. Edwards Deming, después de la Segunda Guerra Mundial, fueron un hito importante.

- Deming destacó que la variabilidad era el mayor adversario de la calidad, y que no solo debía ser administrada a nivel operativo, sino también desde la dirección. Su célebre conjunto de "14 principios de gestión" abogaba por el liderazgo, la formación de los empleados y la visión a largo plazo. Asimismo, subrayaba que la calidad no tenía que estar supeditada a la revisión final, sino al sólido diseño del procedimiento desde el comienzo [4].

- Juran, por su parte, fue pionero en entender la calidad como un problema de gestión y no solo técnico. Introdujo la llamada “trilogía de Juran”: planificación, control y mejora de la calidad. También fue uno de los primeros en destacar los costes de la no calidad, es decir, las pérdidas derivadas de errores, reprocesos y desperdicios [6].

La implementación de estos principios en Japón desde la década de 1950 revolucionó la industria nipona y le permitió convertirse en un referente en términos de calidad y competitividad. El "Premio Deming", establecido en 1951, se transformó en un emblema del compromiso de Japón con la excelencia [4].

De aseguramiento de la calidad a sistemas integrados (TQM, ISO)

Desde la década de 1960 y 1970, la calidad dejó de ser vista como una actividad aislada y pasó a ser considerada como un enfoque integral. El Total Quality Management (TQM) es una filosofía de gestión que surgió a partir de esta evolución (Figura 2). Su objetivo no era únicamente prevenir fallos, sino también promover la mejora continua como cultura corporativa, el enfoque en el cliente y la implicación de todos los estratos organizativos. La noción de que la calidad no se restringe a la producción, sino que abarca todas las funciones de la compañía, desde el diseño hasta el servicio al cliente [7], fue igualmente introducida por TQM.

Figura 2. Áreas de enfoque en TQM [8]

Simultáneamente, las prácticas de gestión de la calidad empezaron a normalizarse por parte de organismos internacionales. La publicación de la familia de normas ISO 9000 en 1987 es el caso más significativo. Estas reglas posibilitaron la unificación de criterios, simplificaron la certificación y aseguraron la confianza en las cadenas de suministro a nivel global. Sus revisiones subsecuentes (ISO 9001:1994, 2000, 2008 y 2015) han progresado desde una perspectiva documental hacia enfoques centrados en el contexto organizacional, los riesgos y la mejora continua [7], [9].

2.2 Control estadístico de procesos (SPC)

Walter A. Shewhart formalizó el control estadístico de procesos en los Bell Telephone Laboratories a inicios de la década del 20. Como método esencial para distinguir entre la variabilidad natural del proceso y las desviaciones generadas por causas especiales.

Shewhart creó la gráfica de control (control chart). Esta disciplina se fundamentó teórica y prácticamente en su obra principal, Economic Control of Quality of Manufactured Product [3]. Las primeras utilizaciones tuvieron lugar en Bell Labs, donde se fabricaban componentes para teléfonos con el objetivo de regular la uniformidad de los procesos eléctricos y de transmisión. Más tarde, el SPC se amplió a las industrias pesadas y a la producción en gran escala, lo cual fue fundamental para la elaboración de equipos militares estandarizados durante la Segunda Guerra Mundial [10].

El Control Estadístico de Procesos (SPC, por sus siglas en inglés) es un grupo de métodos estadísticos que se utilizan para supervisar un proceso productivo (o de servicio) con el fin de detectar variaciones, distinguiendo entre las que son ordinarias (o causas comunes) y las que son provocadas por causas especiales. Estas últimas pueden señalar que el procedimiento está "fuera de control" o requiere intervención. El objetivo de emplear el SPC es garantizar que el proceso sea estable, predecible y capaz de satisfacer los estándares de calidad requeridos [10][11][12].

2.2.1 Variabilidad en el proceso de producción

No existen dos productos o servicios que sean exactamente iguales, ya que los procesos de producción implican numerosas fuentes de variación, incluso si estos procesos se llevan a cabo como se esperaba. Por ejemplo: dos coches de la misma marca y del mismo modelo pueden no ser igual de duraderos aun teniendo las mismas características ya que puede haber variaciones en el proceso de producción, tales como desgaste de herramientas empleadas, habilidad de los operarios, parámetros físicos como temperatura, humedad, etc.

En SPC se toman muestras pequeñas para evaluar la variabilidad del proceso a lo largo del tiempo y se muestran en un gráfico en el que el eje horizontal presenta el orden de las muestras, y el eje vertical señala la frecuencia de cada muestra. Se le llama después de un número significativo de muestras distribución, si el patrón es estable, como se puede ver en la Figura 3 (b). Las

distribuciones pueden variar según lo que las muestras muestran, como se puede ver en la Figura 3(c).

Si el patrón obtenido en la toma de muestras resulta estable y predecible, se dice que el proceso está bajo control. La Figura 3 [d] evidencia que el proceso está bajo control estadístico. No obstante, si surgen razones particulares de variación, la salida del proceso se vuelve impredecible y no sigue un patrón constante a través del tiempo (como ilustra la Figura 3 [e]).

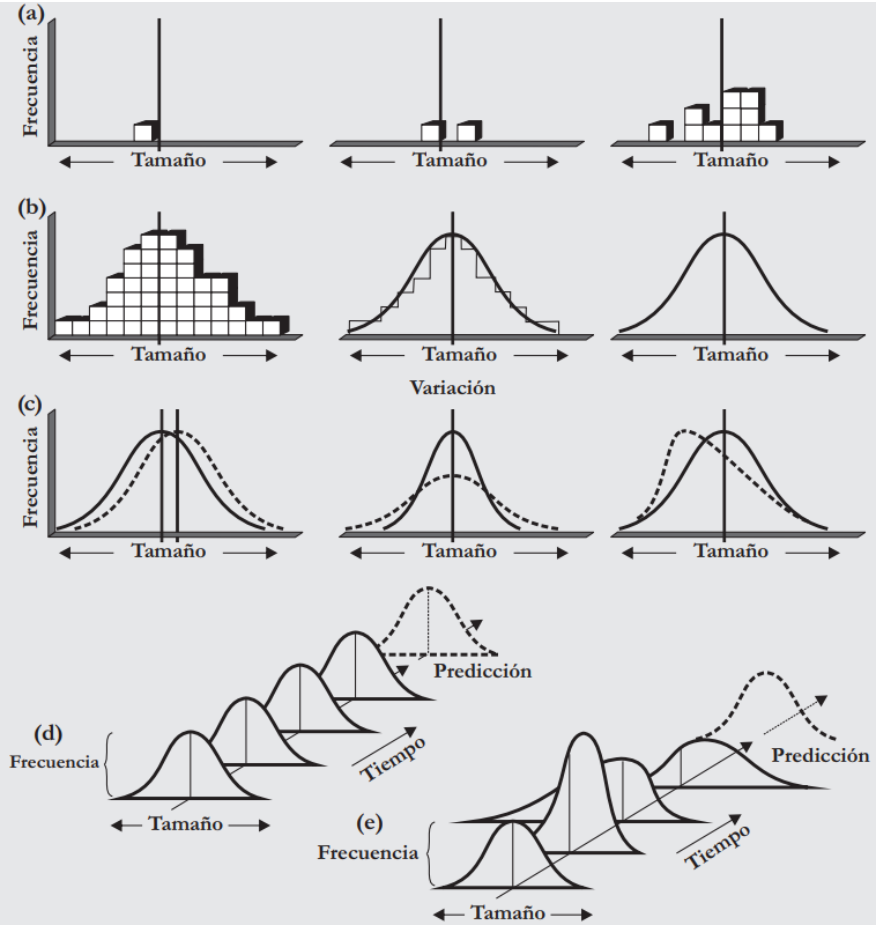


Figura 3. Variabilidad del proceso. (a) Toma de muestras. (b) Distribuciones. (c) Variación de distribuciones. (d) Distribución con causas comunes de variación. (e) Distribución con causas especiales de variación [13]

Este entendimiento es esencial para establecer la previsibilidad y estabilidad del proceso, lo que a su vez orienta la toma de decisiones para optimizar la eficacia y la calidad del proceso de fabricación.

2.2.2 Gráficas de Control

Fueron creadas en los años 20 por Shewhart mientras trabajaba en los laboratorios Bell, estableciendo como condiciones que los datos solo tienen significado dentro de su contexto, es decir en el entorno o aplicación en el cual los estemos midiendo y que para poder extraer información debemos separarlos.

Los gráficos de control de medias ($\bar{\mu}$), desviaciones estándar (σ) o rangos (R) son las principales herramientas del SPC. La definición de subgrupos racionales (n observaciones obtenidas en periodos de tiempo constantes) es el fundamento del método, que posibilita la captura de la variabilidad a corto plazo.

- Media general de subgrupos: media global del proceso a partir de las medias de los subgrupos.

$$\bar{\bar{\mu}} = 1/n \sum_{t=1}^{n} \bar{x_t} \qquad (2.1)$$

- Desviación Estándar:

$$\bar{\sigma} = \frac{1}{n-1} \sum_{t=1}^{n} (x_t - \mu) \qquad (2.2)$$

- Rango del subgrupo: variabilidad dentro de cada subgrupo medida como la diferencia entre el valor mayor y el menor.

$$R_i = max(x_t) - min(x_t) \qquad (2.3)$$

- Media de rangos: variabilidad promedio del proceso a partir de los rangos de todos los subgrupos.

$$\bar{R} = \frac{1}{m} \sum_{i=1}^{m} R_i \qquad (2.4)$$

- Límites de control para gráficas de medias: umbrales superior e inferior que determinan si la media del proceso está bajo control.

$$UCL = \bar{\bar{X}} + A_2\bar{R}, \quad LCL = \bar{\bar{X}} - A_2\bar{R} \tag{2.5}$$

Donde A_2 es un factor que depende del tamaño de muestra n.

Es fundamental saber elegir los límites de control. Si los límites son muy estrechos, se detectarán las variaciones propias del proceso como fallos, lo que provocará alarmas falsas cuando el proceso funcione normalmente; si los límites son demasiado amplios, en cambio, podrían no detectar desviaciones importantes de la operación normal (Figura 4).

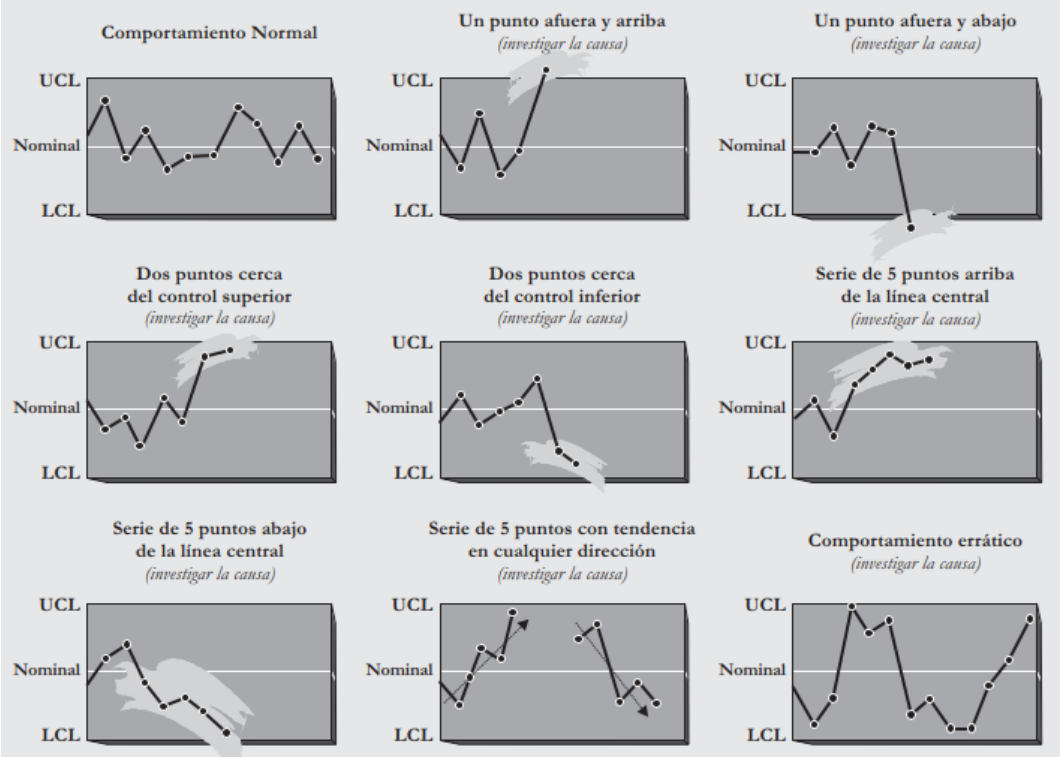


Figura 4. Gráficas de control en diversos tipos de estados, desde situaciones bajo control a comportamiento errático [13]

2.3 Control Estadístico de Procesos Multivariable (MSPC)

El Control Estadístico de Procesos Multivariado (MSPC, por sus siglas en inglés) es la ampliación del SPC tradicional a situaciones en las que es necesario supervisar al mismo tiempo múltiples variables de calidad que tienen correlación entre ellas.

El MSPC toma en cuenta el vector de variables en conjunto, lo que hace posible identificar patrones anómalos que no se encontrarían si las variables se examinaran por separado [14], [15]. En cambio, el SPC univariante estudia una sola característica a la vez. El MSPC fue creado en el periodo de 1940 cuando Harold Hotelling presentó la estadística T^2 que es una generalización multivariante de la t de Student [16].

Durante la década de 1980 y 1990, se expandió el empleo de los estadísticos T^2 y Q (SPE) en la supervisión de procesos químicos, farmacéuticos y manufacturados a gran escala debido a la popularización de métodos para reducir dimensionalidad como PLS y PCA [17] [18]. El MSPC utiliza modelos multivariantes que capturan la correlación entre variables. Entre las herramientas más relevantes se encuentran:

2.3.1 T^2 de Hotelling

La variabilidad de una observación se mide a través del T^2 de Hotelling, ya sea dentro del modelo multivariante o del espacio de componentes principales. Se entiende como una medida de distancia estadística en relación con el centro del modelo.

Si las variables de los conjuntos fuesen independientes, T^2 sería la distancia euclídea cuadrática, pero en la práctica, dado que las variables guardan correlación esta distancia, es la distancia de Mahalanobis, la cual se ajusta según la covarianza entre variables.

La Figura 5 representa como la distancia de Mahalanobis produce un menor número de valores atípicos, este resultado es debido a que esta emplea un elipsoide asimétrico, que depende de la covarianza de las variables.

$$T^2 = x\ P\ S_{\mathrm{k}}^{-1}\ P^{\top} x^{\top} \qquad (2.6)$$

donde:

x = vector de datos medidos
P = matriz de vectores propios reducida
S_k = matriz diagonal con los valores propios reducida.

Tanto P como S_k son de dimensiones k x k siendo este el número de dimensiones del espacio reducido.

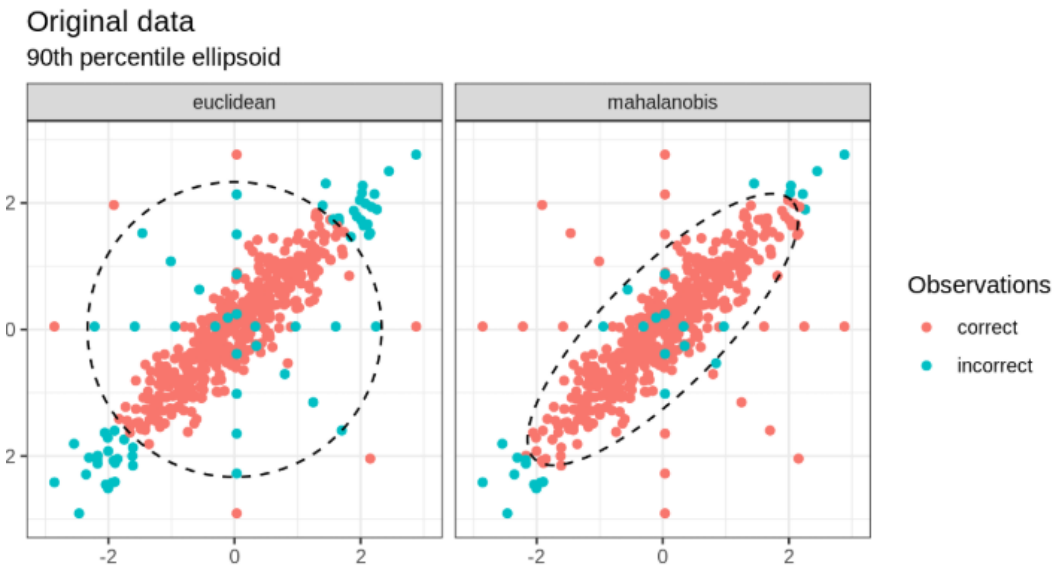


Figura 5. Distancia euclídea frente distancia de Mahalanobis [19]

2.3.2 Error Q (Squared Prediction Error)

La estadística Q, también conocida como SPE (Squared Prediction Error), cuantifica la variabilidad residual de una observación en relación con el modelo (por lo general, un modelo PCA). En otras palabras: qué parte queda fuera del subespacio que explican las componentes principales.

Sea x una observación y $\hat{x}$ su proyección reconstituida:

$$Q = \left|\left|x - \hat{x}\right|\right|^2 = \sum_{j=1}^{p} \left(x_j - \hat{x_j}\right)^2 \qquad (2.7)$$

2.3.3 Uso combinado de T^2 y Q

Un modelo habitual de MSPC emplea los dos gráficos de control al mismo tiempo:

T^2 regula la variabilidad dentro del modelo (en condiciones normales, pero extremas en el ámbito histórico) (Figura 6).

Q regula la variabilidad que no está incluida en el modelo (una nueva variación que no se había previsto en los datos del pasado) (Figura 7).

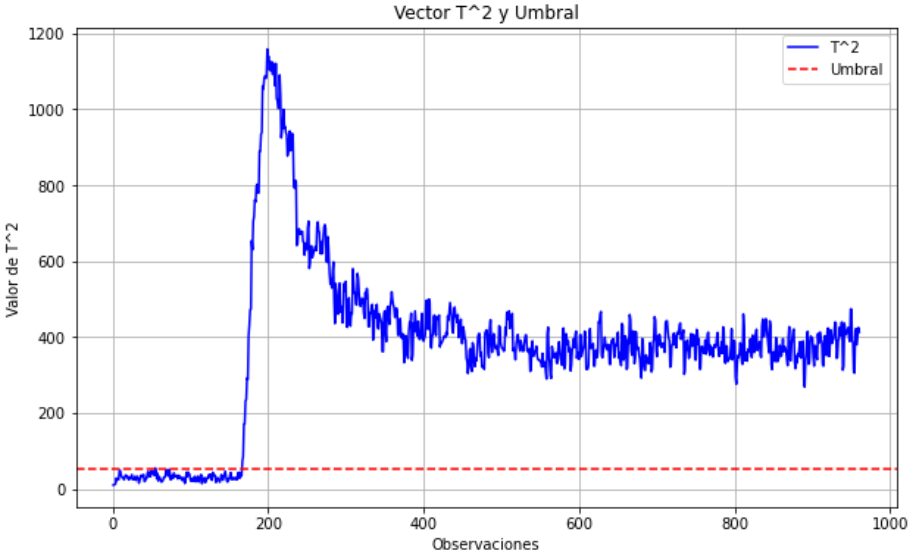


Figura 6. Gráfica de T^2 en una muestra de datos del proceso Tenesee-Eastman con funcionamiento anormal.

MSPC con T^2 y Q se usa en procesos donde se monitorizan múltiples variables correlacionadas, se desea distinguir entre desviaciones extremas pero esperadas (T^2) y nuevos patrones desconocidos (Q) y en los escenarios donde es crucial anticipar fallos, como en procesos químicos, farmacéuticos y de manufactura avanzada [15], [17].

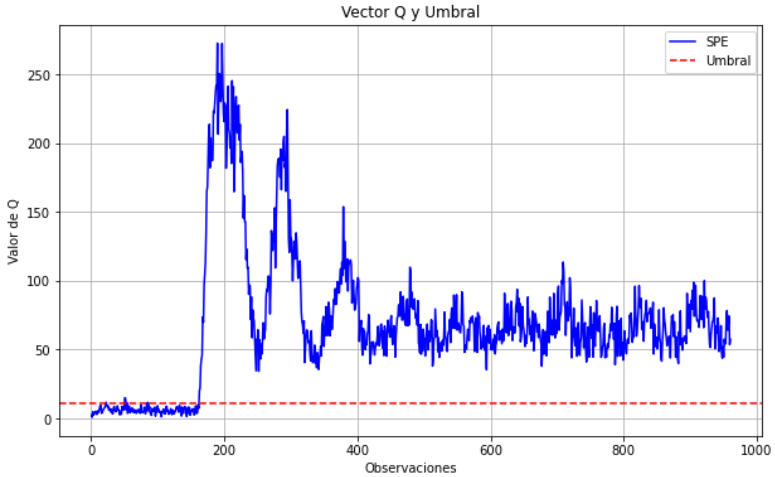


Figura 7. Gráfica de Q, en una muestra de datos del proceso Tenesee-Eastman con funcionamiento anormal

2.4 Análisis de componentes principales (PCA)

El análisis de componentes principales (PCA) fue introducido por Karl Pearson en 1901 como un método geométrico para representar datos en espacios de menor dimensión [20]. Posteriormente, Harold Hotelling (1933) formalizó su marco matemático en términos de álgebra matricial y estadística.

Las primeras aplicaciones tuvieron lugar en el campo de la economía, la biología y la psicología. En los años finales del siglo XX, el PCA se estableció en la supervisión de calidad multivariada, especialmente en MSPC, en la que se emplea para crear modelos de referencia del proceso utilizando información histórica de operación normal [14], [17].

Este tipo de análisis se emplea con los siguientes objetivos:

a. Disminuir la dimensionalidad para poder observar de manera sencilla datos de procesos complejos y detectar vínculos entre variables
b. Desarrollar modelos de referencia para la detección de anomalías y MSPC.
c. Depurar el ruido y optimizar la capacidad predictiva en sistemas multivariantes [15], [17], [18].

Es un modelo frecuentemente empleado en supervisión de procesos industriales químicos (como es el caso de este trabajo), farmacéuticos y petroquímicos, reconocimiento de patrones en visión artificial y en estudios biomédicos y genómica.

El análisis de componentes principales (PCA, por sus siglas en inglés) es un método estadístico multivariante que se emplea para disminuir la dimensionalidad de un conjunto de datos que contiene varias variables correlacionadas.

La idea fundamental es que la varianza de una serie de datos mide la cantidad de información que contiene, matemáticamente, se realiza mediante la descomposición espectral de la matriz de covarianzas o, de forma equivalente, en la descomposición en valores singulares (SVD).

Esta técnica convierte dichas variables en un nuevo conjunto de variables no correlacionadas, conocidas como componentes principales (PCs). Cada componente principal es una combinación lineal de las variables originales y se organiza así:

1. El componente inicial recoge la varianza más alta que los datos pueden ofrecer.
2. El segundo componente, bajo la condición de ser ortogonal al primero, recoge el mayor porcentaje de la varianza restante.
3. Y así consecutivamente [17], [18].

Idealmente, los primeros componentes capturan la mayor cantidad de información y el resto el ruido (Figura 8).

Su formulación matemática empieza tomando una matriz de datos de comportamiento normal del proceso: $X \in R^{n*m}$ con n muestras, y m variables:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \quad (2.8)$$

Normalizamos con media 0 y varianza 1, para obtener la variabilidad de los datos, es decir, a cada valor de la matriz de datos original (X) se le resta la media μ_m y se divide por su desviación estándar σ_m, de su variable, para obtener la matriz de datos normalizados X_n:

$$X_n[m,m] = \frac{X[n,m] - \mu_m}{\sigma_m} \quad (2.9)$$

Calculamos la matriz de covarianzas R:

$$R = \frac{1}{n-1} X_n^T X_n \quad (2.10)$$

Siendo R:

$$R = \begin{pmatrix} 1 & r_{12} & .. & r_{1n} \\ r_{21} & 1 & .. & r_{2n} \\ : & : & . & : \\ r_{m1} & r_{m2} & .. & 1 \end{pmatrix} \quad (2.11)$$

A partir de R realizamos la descomposición en valores singulares (SVD).

$$svd(R) = VSV^T \quad (2.12)$$

De la ecuación (2.12) obtenemos los valores propios S (Figura 8) y los vectores propios (V). La proyección del vector de observaciones $X_n \in R^m$ desacopla el espacio de observaciones en un conjunto T, la columna i-ésima de V es el vector de carga v_i (loadings) que transforma x_n en el score $t_i = x_i V$.

$$T = X_n V \qquad (2.13)$$

Los loadings (V) son los coeficientes de las combinaciones lineales que determinan cada uno de los componentes principales, estos, señalan el grado de contribución de cada variable original a cada uno de los componentes. Por ejemplo, si la primera componente de la variable "temperatura" tiene un loading alto, significa que las fluctuaciones del proceso en dicha componente están fuertemente afectadas por las variaciones de temperatura.

Por otra parte, los scores (T) son la representación de cada observación en los ejes determinados por los componentes principales, en otras palabras, los scores representan las coordenadas renovadas de los datos en el espacio reducido. Posibilitan la detección de agrupaciones, anomalías o tendencias y el análisis de la estructura del proceso.

Una vez obtenida la representación de la matriz de scores (T) reducimos la dimensionalidad del conjunto en base a la variabilidad que deseemos capturar, frecuentemente se emplea como criterio el 90%, dado que capturamos la mayor parte de la información, reducimos significativamente la dimensión del espacio, ya que como hemos comentado anteriormente, dicha variabilidad, se encuentra en los primeros componentes principales.

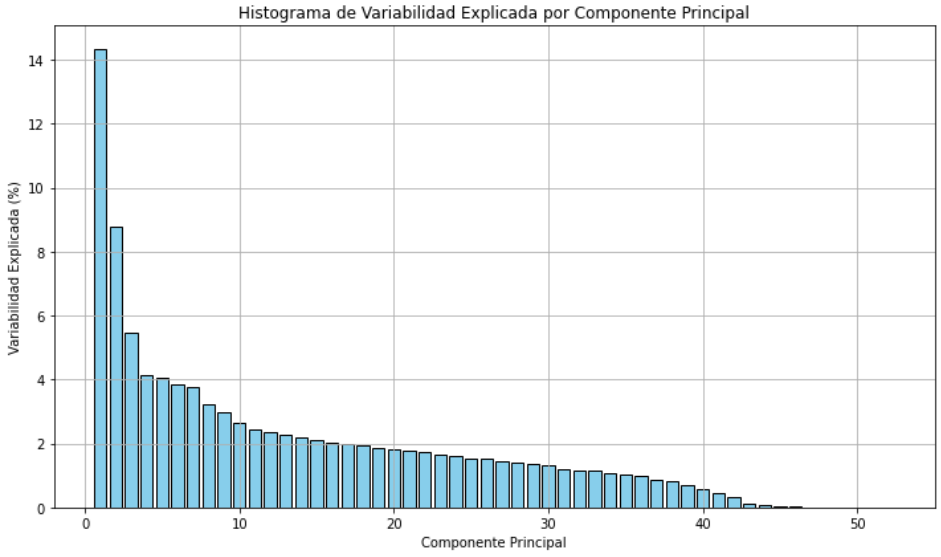


Figura 8. Variabilidad capturada por componente principal en el proceso Tennessee-Eastman

Como

$$S = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ : & : & . & : \\ 0 & 0 & \dots & \lambda_m \end{pmatrix} \tag{2.14}$$

Calculamos el número de componentes que satisfacen la condición:

$$var(\%) = \sum_{1}^{i} \frac{\lambda_i}{tr(S)} \cdot 100 \tag{2.15}$$

Siendo var(%) la variabilidad a capturar expresada en porcentaje.

Reteniendo los a primeros vectores de carga más grandes obtenidos en la ecuación (2.12) tenemos la matriz de autovectores reducida $P(m\ x\ a)$ siendo a el número de dimensiones resultantes.

Ahora podemos representar los scores de dimensionalidad reducida T_a, $(n \times a)$,

$$T_a = X_n\ P \tag{2.16}$$

Al igual que en (2.13) se cumple que $t_i = x_i\ P$.

Entonces ahora podemos aplicar MSPC, y calcular las estadísticas T^2 y Q. T^2 se calcula como se mostró en la ecuación (2.6) y Q lo calcularemos como el producto de la matriz de residuos por su transpuesta, lo cual es equivalente a la diferencia cuadrada de los datos originales y su reconstrucción:

$$Q_i = r_i\ r_i^T \tag{2.17}$$

Siendo

$$r_i = (I - PP^T) \tag{2.18}$$

donde r_i es la fila i-ésima de la matriz de residuo e I es la matriz identidad $m \times m$.

Q, por tanto, mide la diferencia entre una muestra y su proyección al espacio reducido (si fueran iguales el vector de residuos r sería cero). El error de predicción cuadrático Q se genera porque, al realizar la reducción dimensional, hemos descartado parte de la información.

La identificación de una anomalía tipo Q en un sistema de control significa que la correlación entre las variables ha sufrido un cambio importante, lo cual quiere decir que ya no se comportan las variables entre sí como lo hacían durante los datos de funcionamiento habitual [21].

2.5 Redes Neuronales Artificiales (ANN)

Las redes neuronales artificiales (ANN, por sus siglas en inglés) son sistemas computacionales que se basan en el funcionamiento y la estructura del cerebro humano. Se diseñan para identificar patrones y comprender relaciones complejas entre variables (Figura 9).

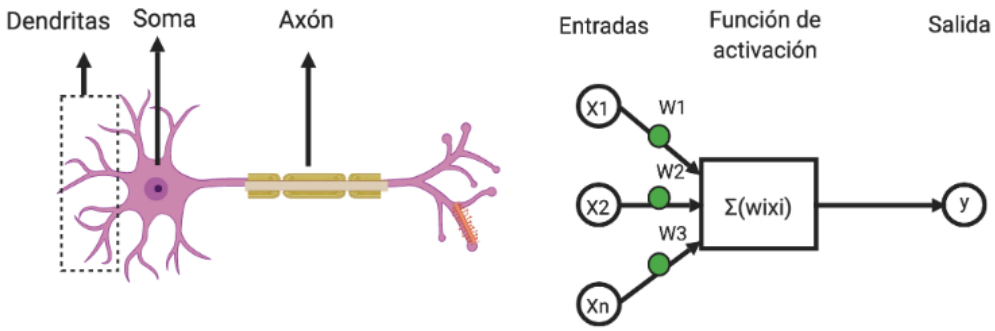


Figura 9. Similitud entre una neurona humana y una neurona artificial

El primer modelo matemático de una neurona artificial fue creado por Walter Pitts y Warren McCulloch (1943) en la mitad del siglo XX, cuando probaron que era capaz de realizar operaciones lógicas sencillas [22]. Más tarde, en 1958, Frank Rosenblatt implementó el Perceptrón, que es visto como la primera red neuronal que puede ser entrenada a través del aprendizaje supervisado [23].

Las redes neuronales despertaron menos interés en las décadas de 1960 y 1970 debido a restricciones teóricas y computacionales, pero volvieron a ser relevantes en la década de 1980 con el desarrollo del algoritmo de retropropagación del error (backpropagation), hito que permitió sentar las bases del aprendizaje automático sin un coste computacional excesivo [24].

Hoy en día, las redes neuronales han progresado hasta arquitecturas profundas (deep learning) gracias al aumento de la capacidad computacional, a la posibilidad de acceder a enormes cantidades de datos (big data) y al empleo de unidades de procesamiento gráfico (GPU). Estas arquitecturas son

capaces de llevar a cabo trabajos complejos como el control de procesos industriales, la predicción y el diagnóstico de errores, el procesamiento del lenguaje natural y la visión artificial.

2.5.1 Neurona artificial

La Neurona Artificial (Figura 10) es la unidad elemental de procesamiento en una red neuronal. Su tarea es recibir un grupo de entradas (inputs), ponderarlas a través de pesos (weights), sumar los resultados y, para conseguir una salida (output), usar una función de activación.

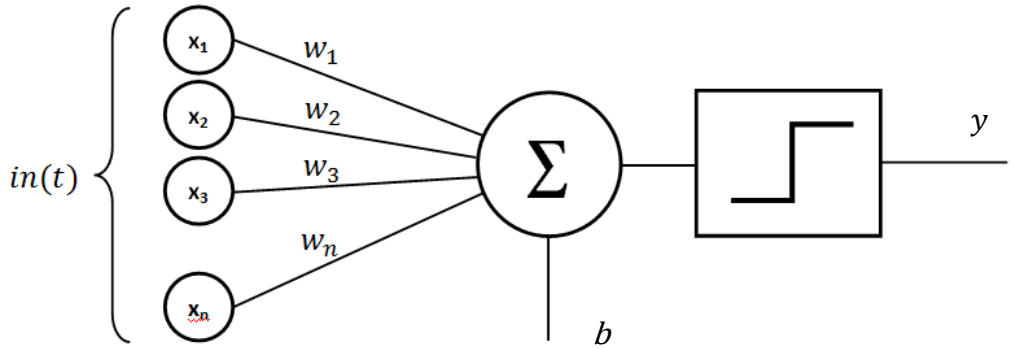


Figura 10. Esquema de la unidad neuronal (Perceptrón)

El modelo matemático general de una neurona se expresa como:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.19}$$

donde:

- x_i: son las entradas o variables de entrada,
- w_i: son los pesos sinápticos que representan la fuerza de conexión entre neuronas,
- b: es el término de sesgo (bias), que permite desplazar la función de activación,
- $f(\cdot)$: es la función de activación
- y: es la salida de la neurona.

Este modelo permite representar relaciones lineales y no lineales dependiendo de la forma de $f(\cdot)$.

2.5.2 Red de neuronas

Un grupo de neuronas dispuestas en capas es lo que se denomina red neuronal (Figura 11):

1. Capa de entrada: recibe los parámetros de entrada o las características del problema.
2. Capas ocultas: Analizan la información utilizando combinaciones lineales y funciones de activación que no son lineales.
3. Capa de salida: Produce las clasificaciones o predicciones finales.

La configuración fundamental de una red multicapa (Multi-Layer Perceptron, MLP) puede representarse en forma matricial como:

$$y = f^{(L)}\left(W^{(L)} f^{(L-1)}\left(W^{(L-1)} \dots f^{(1)}\left(W^{(1)} x + b\right) \dots + b^{(L-1)}\right) + b^{(L)}\right) \qquad (2.20)$$

donde L es el número de capas, $W^{(l)}$ las matrices de pesos en la capa l, $b^{(l)}$ los bias en la capa l y $f^{(l)}$ las funciones de activación correspondientes.

Es fundamental incorporar funciones de activación no lineales entre las capas, porque esto le da a la red la posibilidad de aproximar funciones que no son lineales y aprender relaciones complejas entre las variables de entrada y salida.

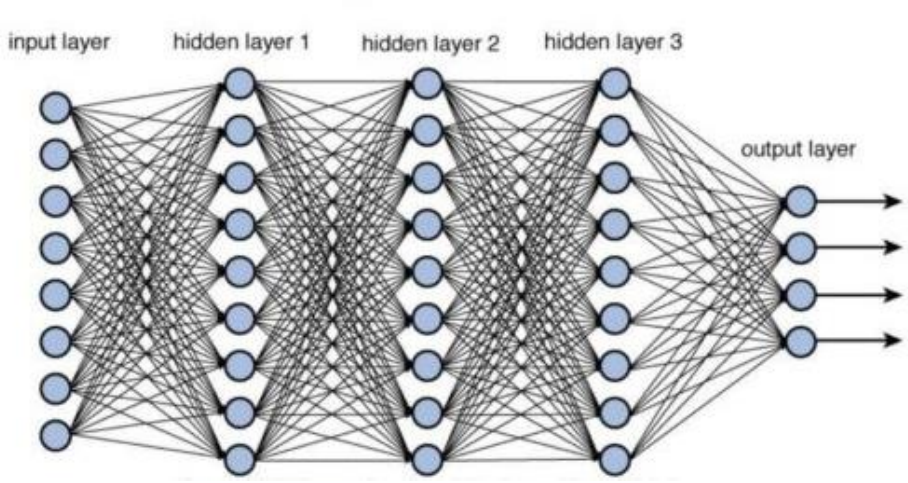


Figura 11. Red Neuronal MLP de 5 capas

2.5.3 Funciones de activación

La salida de cada neurona está determinada por las funciones de activación en relación a la suma ponderada de sus entradas. Incorporan no linealidad al

modelo, lo que posibilita que las redes adquieran patrones complejos. De lo contrario, la red completa se podría colapsar en una sola recta, plano o hiperplano, dependiendo de las dimensiones, debido a su linealidad.

Estudiar qué tipo de función emplear para cada aplicación es fundamental, a continuación, algunas de las más destacadas y empleadas en este trabajo:

1. **Función sigmoide:** hace que los valores grandes saturen en 1 y pequeños en 0 (Figura 12), pero en este caso las derivadas no son nulas

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.21}$$

   Esta función en valores cercanos a 0 tiene muy buena inclinación, sin embargo, con valores muy grandes o muy pequeños es casi horizontal, es decir, la derivada se hace muy pequeña.

   En backpropagation (algoritmo de entrenamiento, mencionado más adelante) con derivadas tan pequeñas, los ajustes a pesos y sesgos serán muy pequeños entonces, por lo tanto, la red aprenderá despacio o dejará de aprender, esto es lo que se conoce como desvanecimiento del gradiente.

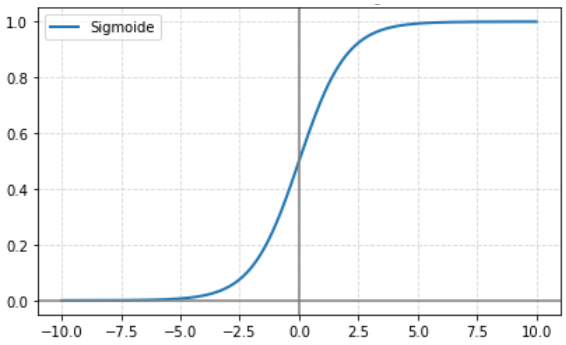


Figura 12. Función sigmoiode

2. **Tangente hiperbólica** (tanh) (Figura 13): Genera valores entre -1 y 1. Tiende a converger más rápido que la sigmoide al tener salidas centradas en cero.

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.22}$$

En la mayoría de casos es mejor que la sigmoide, dado que, está centrada en el 0, puede tomar valores positivos y negativos, solucionando el problema de la sigmoide y su derivada es mayor por lo tanto el aprendizaje será más rápido.

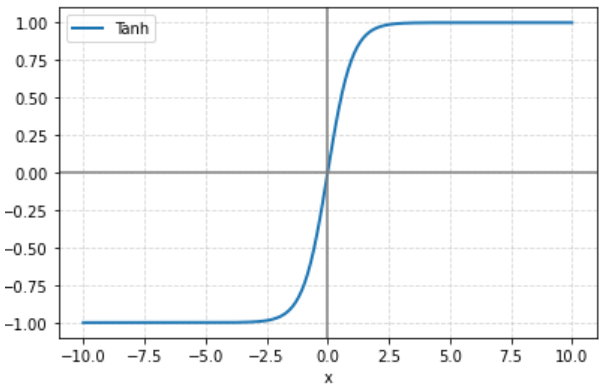


Figura 13. Función tangente hiperbólica

Aun así, presenta el problema de desvanecimiento de gradiente y tiene un alto coste computacional.

3. **Unidad Rectificada Lineal** (ReLU) (Figura 14): Es una función extremadamente simple, no consta de exponentes ni, cálculos, simplemente devuelve lo que sea mayor:

$$ReLU(x) = \max(0, x) \qquad (2.23)$$

En un estudio de 2011 se reveló que es mejor en la mayoría de casos que la tanh, observándose velocidades de backpropagation y de propagación hacia adelante 6 veces mayor [25].

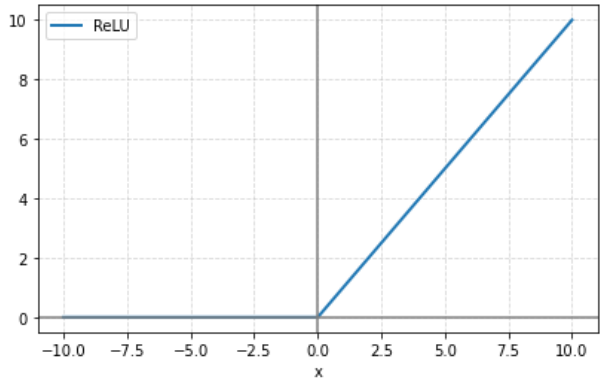


Figura 14. Función ReLU

Esta función, en la mayoría de aplicaciones es superior a las dos anteriores, tiene un coste computacional muy bajo, su derivada es muy simple y no está acotada para números positivos (al contrario que $\sigma(x)$ y $\tanh(x)$) lo que genera un gradiente constante y un aprendizaje más rápido en la mayoría de casos.

Los inconvenientes que tiene, residen en la primera parte de la función, la cual devuelve 0 para todo valor menor que 0, lo cual puede incurrir en que algunas neuronas durante las primeras fases de entrenamiento almacenen un 0 entorpeciendo asi el entrenamiento de la red, la siguiente función suple dichas carencias.

4. **Función Leaky Relu**: Similar a ReLu pero en vez de devolver 0 para números negativos, devuelve un valor muy pequeño, si lo parametrizamos correctamente, por lo general $\alpha < 0{,}1$ (Figura 15)

$$f(x) = \begin{cases} x & si \quad x > 0 \\ \alpha x & si \quad x \leq 0 \end{cases} \tag{2.24}$$

De esta manera tenemos una función no acotada, simple de calcular y solventamos el problema de las neuronas muertas.

Cada función presenta beneficios y restricciones dependiendo de la arquitectura utilizada y el tipo de problema [26], [27].

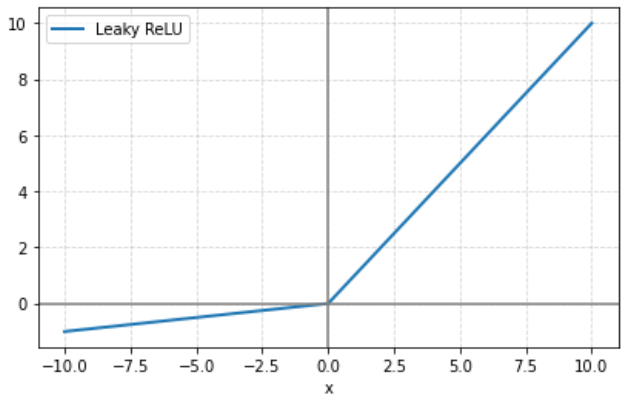


Figura 15. Función Leaky Relu

2.5.4 Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes (RNN, por su nombre en inglés) (Figura 16) son una ampliación de las redes neuronales artificiales convencionales,

creadas con el propósito de manejar datos que dependen del tiempo o son secuenciales, en los que el orden de las observaciones tiene importancia.

A diferencia de las redes multicapa (MLP), en las cuales las entradas y salidas se procesan de manera independiente, las RNN incluyen conexiones de retroalimentación que posibilitan conservar información de pasos anteriores, lo que les proporciona una memoria dinámica del sistema [27], [28].

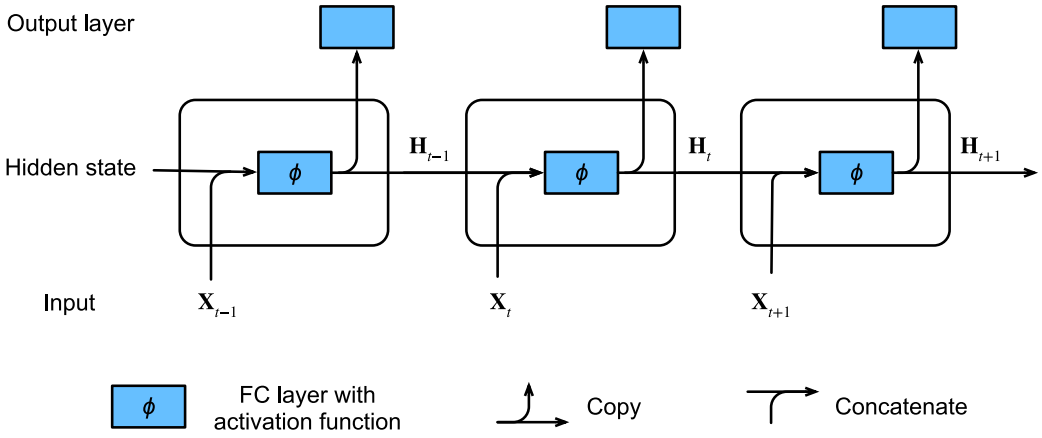


Figura 16. Red neuronal recurrente [29]

El principio básico de una RNN consiste en mantener un **estado oculto** h_t que resume la información relevante de todas las entradas anteriores. En cada instante temporal t, el estado oculto se actualiza en función de la entrada actual x_t y del estado anterior h_{t-1}, de acuerdo con las siguientes ecuaciones:

$$h_t = f_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.25)$$

$$\hat{y}_t = f_y(W_{hy}h_t + b_y) \quad (2.26)$$

Donde x_t representa la entrada en el instante t, h_t es el estado oculto, que almacena la información de contexto, $\hat{y}_t$ es la salida estimada, W_{xh}, W_{hh}, W_{hy} son las matrices de pesos correspondientes a las conexiones de entada recurrente y salida, b_h, b_y son los sesgos y f_h, f_y son las funciones de activación, comúnmente tanh o ReLU.

Se puede observar esta formulación como una red "extendida en el tiempo" (unrolled network), donde cada intervalo de tiempo es igual a una capa que

tiene los mismos parámetros. Este despliegue, en el transcurso del entrenamiento, posibilita que se use el algoritmo de retropropagación a través del tiempo (BPTT) para modificar los sesgos y pesos [27].

2.6 Aprendizaje automático

Con el propósito de reducir la función de coste que evalúa la discrepancia entre la salida estimada y la esperada (ecu. 2.28), el procedimiento para entrenar una red neuronal artificial se fundamenta en optimizar de manera iterativa sus parámetros internos, es decir, sus sesgos (b) y pesos (w). Este proceso se compone de dos etapas fundamentales: la retropropagación del error (backpropagation) y la actualización de parámetros a través de un algoritmo de descenso por gradiente [28].

2.6.1 Propagación hacia adelante (Forward propagation)

En cada época de entrenamiento, la red lleva a cabo inicialmente una propagación hacia adelante (forward propagation), en la que los datos de entrada se convierten capa a capa mediante la aplicación de una función de activación no lineal seguida de una combinación lineal de pesos y sesgos [27]. En términos matemáticos, para la capa l:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}, \qquad a^{(l)} = f^{(l)}\left(z^{(l)}\right) \tag{2.27}$$

donde $a^{(l-1)}$ representa la salida de la capa anterior, $f^{(l)}$ es la función de activación y $z^{(l)}$ la suma ponderada. El resultado final $a^{(L)}$ se compara con la etiqueta real y, mediante una función de coste: $C\left(a^{(L)}, y\right)$, la cual cuantifica el error de predicción [26].

2.6.2 Cálculo del error

Para un conjunto de datos con salidas esperadas y, se define una función de coste C, por ejemplo, el error cuadrático medio (MSE):

$$C = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \tag{2.28}$$

Donde m es el número de muestras, y_i es el valor real e $\hat{y}_i$ es la predicción del modelo. En forma matricial, considerando la salida de la red $A^{(L)}$:

$$C = \frac{1}{2N} \|Y - A^{(L)}\|_F^2 \tag{2.29}$$

2.6.3 Retropropagación del error

Tras calcular el coste, evaluamos como varía respecto a cada parámetro del modelo (pesos y sesgos). Para ello se aplica la regla de la cadena, definiendo el termino de error local $\delta^{(L)}$:

$$\delta^{(L)} = \frac{\partial C}{\partial a^{(L)}} \odot f'^{(L)}\left(z^{(L)}\right) \tag{2.30}$$

donde $\odot$ representa el producto elemento por elemento.

El error atribuido a cada neurona en la capa de salida está representado por este valor. Para las capas ocultas, el error se transmite de manera inversa:

$$\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \odot f'^{(l)}\left(z^{(L)}\right) \tag{2.31}$$

Una vez obtenido el error de cada capa, los gradientes de la función de coste con respecto a los pesos y sesgos se calculan como:

$$\frac{\partial C}{\partial W^{(l)}} = \delta^{(l)}\left(a^{(l-1)}\right)^T \quad ; \quad \frac{\partial C}{\partial b^{(l)}} = \delta^{(l)} \tag{2.32}$$

Este procedimiento propaga el error desde la capa de salida hasta las capas previas, lo que hace posible modificar los parámetros de manera proporcional a su aporte al error total.

2.6.4 Actualización de parámetros (Gradient Descent)

Estos se actualizan en la dirección opuesta a la que apunta el gradiente, dado que este apunta en la dirección en donde aumenta el error:

$$W^{(l)}(t+1) = W^{(l)}(t) - \eta \frac{\partial C}{\partial W^{(l)}} \; ; \tag{2.33}$$

$$b^{(l)}(t+1) = b^{(l)}(t) - \eta \frac{\partial C}{\partial b^{(l)}} \; ; \qquad (2.34)$$

donde η es la tasa de aprendizaje (learning rate), que controla la magnitud de los ajustes en cada iteración.

De esta forma, la red ajusta sus parámetros en cada iteración buscando minimizar la función de coste de forma incremental. Este proceso se repite para múltiples iteraciones en todo el conjunto de la red, estas iteraciones se denominan épocas (epochs), hasta que el error (los del modelo) converge a un valor aceptable o no mejora.

Existen mejoras en el algoritmo de descenso del gradiente, como el descenso estocástico del gradiente (SGD), Momentum, RMSProp o Adam. Estos son algoritmos avanzados que pueden optimizar este proceso, ya que modifican de manera dinámica la dirección y el tamaño de los pasos de actualización con el fin de prevenir mínimos locales y acelerar la convergencia [27], [28].

2.6.4.1 Adam (Adaptive Moment Estimation)

El algoritmo Adam, empleado en este trabajo, combina las ventajas de Momentum y RMSProp, manteniendo promedios móviles de los gradientes y sus cuadrados para cada parámetro θ (ya sean w o b):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad ; \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \qquad (2.35)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad ; \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad (2.36)$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \qquad (2.37)$$

Donde $g_t = \frac{\partial C}{\partial \theta_t}$ es el gradiente en el momento t, β_1 y β_2 son factores de decaimiento, normalmente 0.9 y 0.999, respectivamente; y ϵ es un término para la estabilidad numérica.

Así, Adam ajusta la tasa de aprendizaje individual de cada parámetro y consigue una convergencia más rápida y estable, lo que lo hace particularmente efectivo para arquitecturas profundas o autoencoders. Adam se ha posicionado como uno de los algoritmos de optimización más utilizados

en el aprendizaje profundo debido a estas características, particularmente en modelos autoencoder y redes convolucionales [25], [27], [28].

2.7 Autoencoders

Las arquitecturas neuronales que pueden aprender representaciones de datos cada vez más compactas y significativas han sido creadas gracias al avance del aprendizaje profundo. Los autoencoders, entre ellos, se caracterizan por su habilidad para codificar información de manera no supervisada, disminuyendo la dimensionalidad del conjunto de datos a la vez que mantienen sus rasgos fundamentales [27].

En la década de 1980, los autoencoders aparecieron como una extensión de las redes neuronales feedforward, con la finalidad de adquirir una representación interna (latente) de los datos sin requerir etiquetas. El objetivo inicial era replicar métodos de reducción de dimensionalidad lineales, como el Análisis de Componentes Principales (PCA), aunque a través de una formulación que no fuera lineal y adaptable [28].

A través de la demostración de su efectividad en la preinicialización de redes profundas, Geoffrey Hinton y sus colegas reanimaron el uso de autoencoders en los años 2000, lo que posibilitó entrenar modelos que previamente eran inestables debido al elevado número de parámetros [27].

Los autoencoders (Figura 17) se convirtieron en una herramienta fundamental para la detección de anomalías, el aprendizaje de características, la reconstrucción de datos y la compresión gracias al avance del deep learning. Su uso abarca diversos campos, desde series temporales industriales hasta visión artificial.

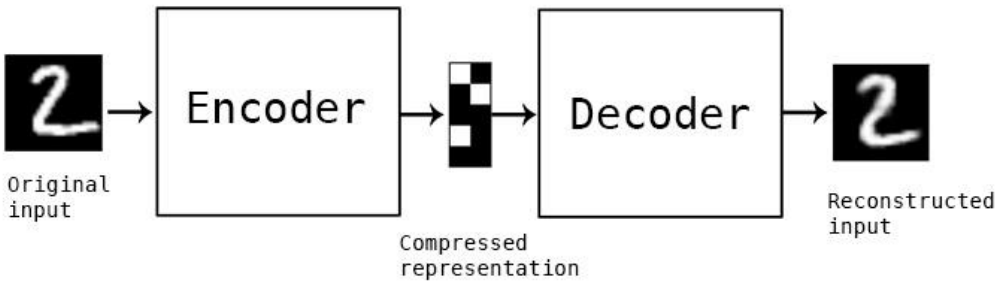


Figura 17. Esquema de un autoencoder

En términos generales, un autoencoder es una red neuronal que ha sido entrenada para replicar su entrada en la salida; sin embargo, tiene como limitación estructural la necesidad de aprender una representación comprimida y significativa del conjunto de datos. Su arquitectura está constituida por dos componentes fundamentales:

- Codificador (Encoder): transforma la entrada $X \in \mathbb{R}^n$ en una representación latente $h \in \mathbb{R}^k$, donde $k < n$.

- Decodificador (Decoder): intenta reconstruir la entrada original a partir de la representación latente, produciendo una salida $\hat{X}$.

Matemáticamente, este proceso se expresa como:

$$h = f_{enc}(W_e X + b_e) \tag{2.38}$$

$$\hat{X} = f_{dec}(W_d h + b_d) \tag{2.39}$$

donde f_{enc} y f_{dec} son funciones de activación no lineales, W_e, W_d son las matrices de pesos del codificador y decodificador y b_e, b_d sus correspondientes sesgos.

El entrenamiento se realiza minimizando una función de coste que mide la diferencia entre la entrada y su reconstrucción. Comúnmente se utiliza el *Error Cuadrático Medio (MSE)*:

$$C = \frac{1}{N} \sum_{i=1}^{N} \left(X_i - \hat{X}_i\right)^2 \tag{2.40}$$

Los métodos mencionados en la sección 2.6 Aprendizaje automático, es decir, la retropropagación del error, la propagación hacia adelante y la actualización de los parámetros a través de algoritmos de descenso del gradiente (normalmente optimizados con Adam [27], [28]), se utilizan para minimizar esta función.

La capa intermedia (h), que también se llama capa latente o cuello de botella, es el núcleo del modelo porque allí reside la representación comprimida de los datos originales. La habilidad de un autoencoder para sobreajustar o generalizar la información depende del tamaño y la estructura de este.

Durante años, se han creado numerosas versiones de autoencoders que están diseñadas para una variedad de tipos de datos y metas de aprendizaje. Se incluyen entre las más significativas:

- Autoencoder totalmente conectado (MLP): está formado solamente por capas densas. Este es el modelo base y se utiliza sobre todo para tablas de datos o vectores de características.

- Autoencoder convolucional (CAE): Se emplean capas convolucionales para capturar estructuras espaciales, y su uso es bastante común en imágenes y visión por computador.

- Autoencoder recurrente (RAE / LSTM): creado para trabajar con secuencias temporales, que introduce dependencias a lo largo del tiempo utilizando unidades GRU o LSTM.

- Autoencoder variacional (VAE): en vez de aprender una codificación determinista, emplea una formulación probabilística y se enfoca en el aprendizaje de una distribución latente.

- Autoencoder con ruido (Denoising AE): adiestrado para reconstruir la entrada original a partir de versiones dañadas o con mucho ruido, lo que aumenta su resistencia ante alteraciones.

- Autoencoder disperso (Sparse AE): establece limitaciones de activación para fomentar representaciones latentes más comprensibles y específicas.

- Autoencoder contractivo (Contractive AE): favorece una codificación más estable al sancionar la sensibilidad del espacio latente a mínimas variaciones en la entrada.

En este trabajo se han implementado dos tipos de autoencoders con finalidades distintas:

1. Autoencoders densos (Fully Connected Autoencoders), que son apropiados para la reconstrucción de variables instantáneas o estáticas.

2. Autoencoders LSTM, que poseen memoria y tienen la habilidad de modelar las dependencias temporales en los datos procesados.

Las dos arquitecturas se analizan detalladamente en los siguientes subapartados.

2.7.1 Autoencoder Denso (Fully Connected Autoencoder)

Los autoencoders densos (Figura 18), que también son conocidos como fully connected o vanilla autoencoders, representan el tipo más básico y común de autoencoder. Su estructura se fundamenta en una red neuronal de tipo feedforward que cuenta con capas totalmente interconectadas (Dense layers); en estas, cada neurona de una capa se enlaza con todas las neuronas de la capa que le sigue [27], [28].

Cuando no hay dependencias temporales o espaciales, por ejemplo, en datos tabulares o en vectores de características estáticas, este tipo de arquitectura es particularmente adecuada. Al estar diseñado para reconocer relaciones no lineales entre las variables, es una herramienta útil para la detección de anomalías, la reconstrucción de procesos industriales y la reducción de dimensionalidad. Conceptualmente es similar al PCA, aunque con una formulación no lineal.

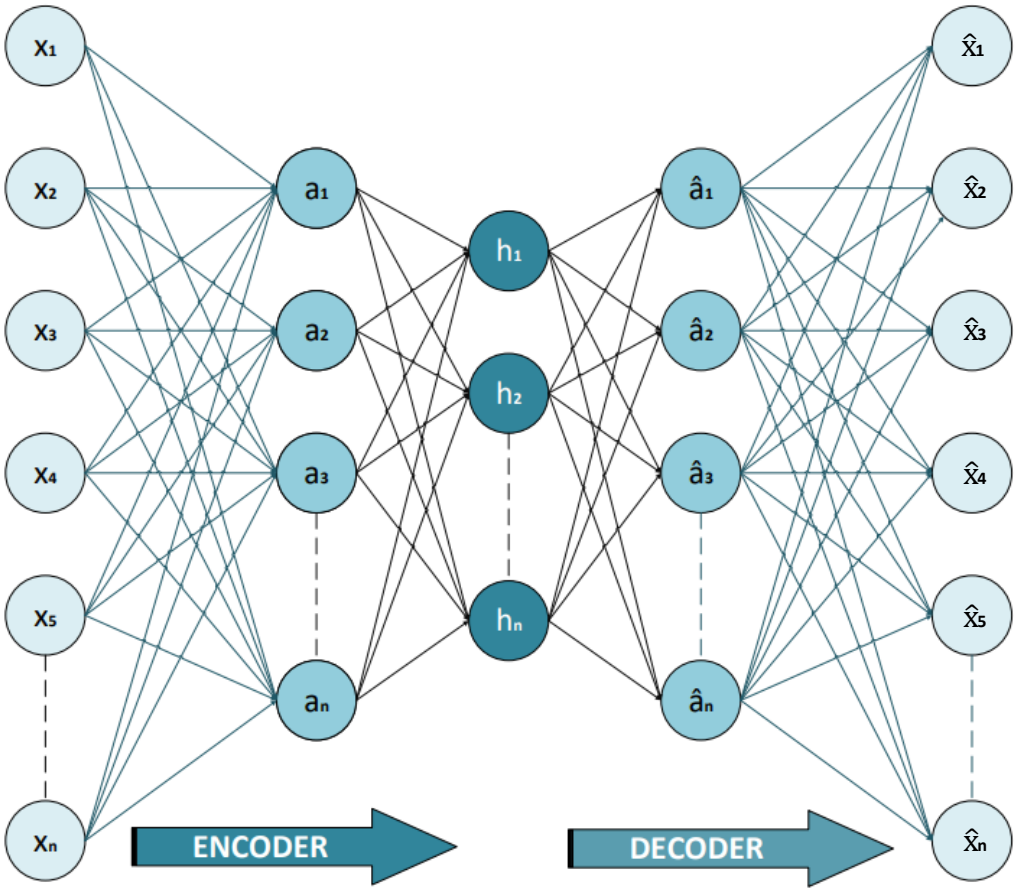


Figura 18. Representación interna de un autoencoder de 5 capas y n neuronas

En este trabajo, se utilizan los autoencoders densos como fundamento para la reconstrucción de datos estáticos normalizados, sirviendo como modelo de referencia en comparación con arquitecturas más sofisticadas, por ejemplo, las LSTM.

El modelo se compone de tres bloques principales:

1. **Codificador:** transforma la entrada $X \in \mathbb{R}^n$ en una representación latente $h \in \mathbb{R}^n$ a través de sucesivas combinaciones lineales y funciones de activación no lineales:

$$h = f_{enc}(W_e X + b_e) \tag{2.41}$$

   donde $W_e \in \mathbb{R}^{n \times k}$ son los pesos del codificador y $f_{enc}(\cdot)$ suele ser una función *ReLU* o *LeakyReLU* que introduce no linealidad al proceso [25].

2. **Capa latente:** actúa como cuello de botella, limitando la cantidad de información que puede fluir a través del modelo. Esta restricción obliga a la red a extraer las características más relevantes de los datos de entrada, eliminando redundancias.

3. **Decodificador:** reconstruye la entrada original a partir del vector latente:

$$\hat{X} = f_{dec}(W_d h + b_d) \tag{2.42}$$

   donde $W_d \in \mathbb{R}^{n \times k}$ son los pesos del codificador y $f_{enc}(\cdot)$ puede ser una función *ReLU* o sigmoide dependiendo de la naturaleza de los datos de salida.

El propósito del entrenamiento es reducir la brecha entre la entrada (X) y su reconstrucción $\hat{X}$, sin sobreajuste, a través de una función de coste (C).

En el caso de este trabajo se ha usado el error cuadrático medio MSE, el cual se expresa de manera matricial de la siguiente manera:

$$C = \frac{1}{2N} \ \|X - \hat{X}\|_F^2 \tag{2.43}$$

El entrenamiento sigue el mismo procedimiento descrito en el apartado 2.6 Aprendizaje automático.

2.7.2 Autoencoder Recurrente (RAE) variante LSTM (Long Short-Term Memory Autoencoder)

Redes LSTM

Las redes neuronales LSTM son una evolución de las Redes neuronales Recurrentes (RNN), estas surgieron como extensión de las redes feedforward para modelar datos secuenciales, permitiendo que la salida de una unidad en un paso temporal influya como entrada en el paso siguiente. Las RNN convencionales pueden propagar información temporalmente, pero sufren el problema del desvanecimiento del *gradiente* o *explosivo* al tratar secuencias largas [29].

Sepp Hochreiter y Jürgen Schmidhuber, en su trabajo "Long Short-Term Memory", presentaron la arquitectura Long Short-Term Memory (LSTM) en 1997. En esta, sugieren una célula con mecanismos de puerta (Figura 19) que controlan el flujo de información y posibilitan que las dependencias a largo plazo se mantengan sin que los gradientes se "apaguen" o "exploten" [30].

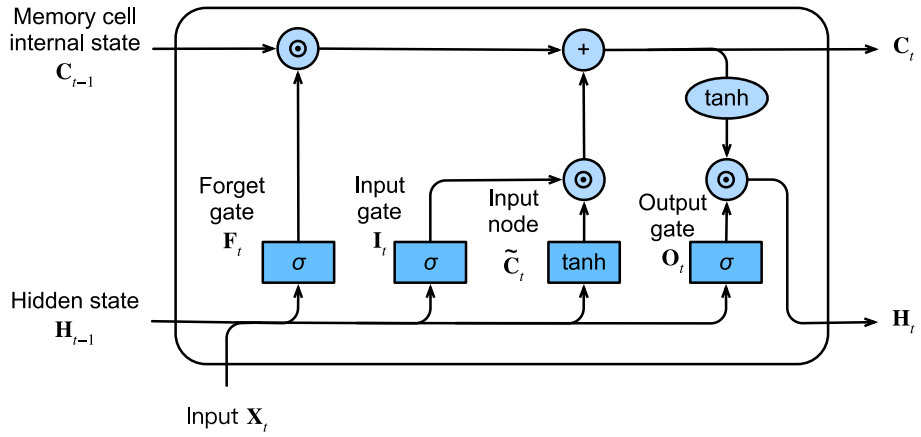


Figura 19. Estructura de una célula LSTM [29]

Una célula LSTM está configurada para administrar y regular el estado interno a lo largo del tiempo, utilizando puertas que determinan qué información se descarta, cuál se incorpora de nuevo y cuál se emite como salida, para ello se define una arquitectura de célula con 3 partes fundamentales, tal y como se ve en la Figura 19.

La primera es la **puerta de olvido** f_t (forget gate), la cual decide qué proporción del estado anterior c_{t-1} conservar.

La segunda es la **puerta de entrada** i_t(input gate) quien regula cuánta parte del candidato $\tilde{c}_t$ se incorpora al nuevo estado de celda, este candidato representa la información propuesta para la memoria, transformado mediante tanh, para mantener valores centrados en $[-1,1]$, de este modo la actualización del estado de celda c_t combina lo retenido y lo nuevo.

En último lugar encontramos la **puerta de salida** o_t (output gate) quien controla qué parte del estado de la celda pasa a la salida h_t filtrándolo de nuevo con la función de activación tanh. Finalmente, h_t constituye la salida oculta de la celda para el paso t.

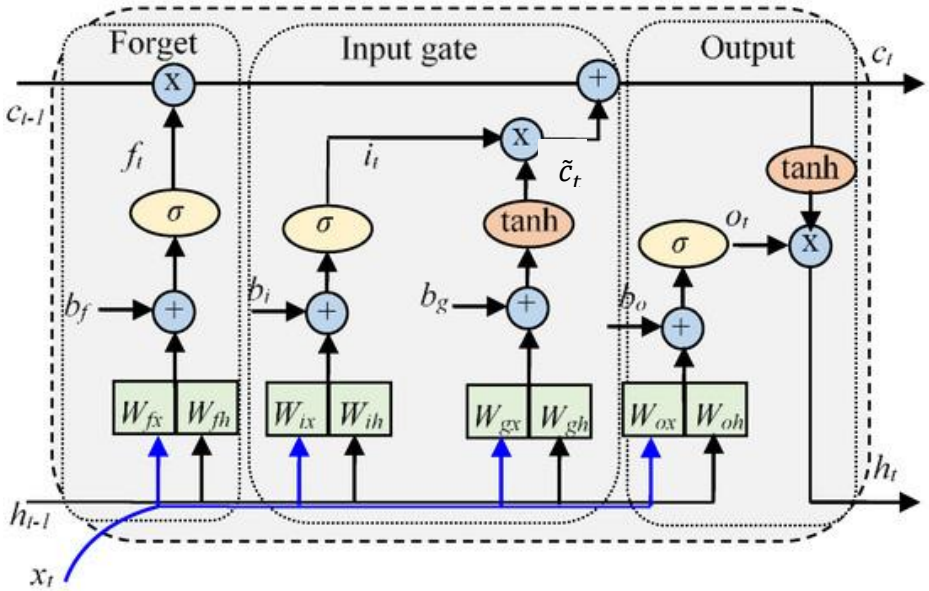


Figura 20. Estructura interna de una unidad LSTM [29]

Las ecuaciones estándar de una célula LSTM (Figura 20), para cada instante t, son:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.44)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.45)$$

$$\tilde{c_t} = \tanh\,(W_c[h_{t-1}, x_t] + b_c) \quad (2.46)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (2.47)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.48)$$

$$h_t = o_t \odot \tanh\,(c_t) \quad (2.49)$$

Siendo:
x_t: Vector de entrada en el instante t.

h_{t-1}: estado oculto del paso anterior.
c_{t-1}: estado de la celda en el paso anterior.
f_t : *forget gate* (puerta de olvido).
i_t : *input gate* (puerta de entrada).
$\tilde{c}_t$: candidato a nuevo estado de la celda.
c_t: estado de la celda actualizado.
o_t: *output gate* (puerta de salida).
h_t: estado oculto de la salida en el paso t.

Como resultado de estas operaciones, una red LSTM permite que una célula LSTM decida cuándo y cuanto recordar, emitiendo la información necesaria y evitando pérdidas, lo cual le dota de una flexibilidad excelente frente a secuencias largas o patrones complejos

Integración de las células LSTM en un Autoencoder

Los autoencoders LSTM (Figura 21) surgen de la combinación entre la arquitectura de los autoencoders clásicos y las redes LSTM, con el propósito de capturar dependencias temporales o secuenciales en los datos. Mientras que un autoencoder convencional se compone de capas densas (*fully connected layers*) que comprimen y reconstruyen representaciones estáticas, los autoencoders LSTM operan sobre secuencias de datos $x_1, x_2, \dots, x_T$ lo que los hace adecuados para aplicaciones donde el orden temporal es esencial, como series de tiempo, señales de sensores o texto [29].

El principio de funcionamiento se mantiene:

- Un encoder LSTM procesa la secuencia de entrada y resume su información en una representación latente (estado oculto final h_T o estado de celda c_T).
- Un decoder LSTM toma esta representación comprimida y genera una reconstrucción de la secuencia original, tratando de minimizar la diferencia con la entrada.

La estructura general de un Autoencoder puede representarse como:

$$Encoder: (h_t, c_t) = LSTM_{enc}(x_t, h_{t-1}, c_{t-1}) \quad (2.50)$$

$$Decoder: (\hat{h}_t, \hat{c}_t) = LSTM_{dec}(y_t, \hat{h}_{t-1}, \hat{c}_{t-1}) \quad (2.51)$$

$$Salida: \hat{y}_t = W_o\, \hat{h}_t + b_o \quad (2.52)$$

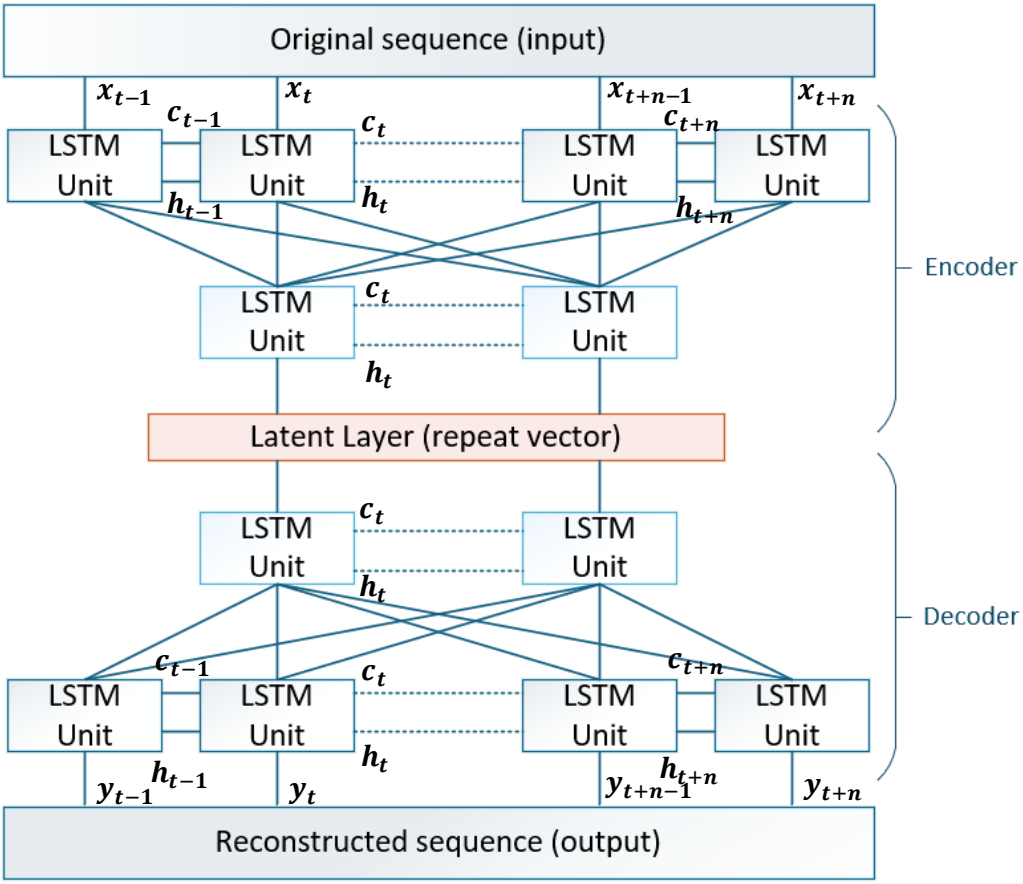


Figura 21. Esquema de un autoencoder recurrente con capas LSTM

El modelo, en el periodo de codificación, pasa por la secuencia de entrada y guarda los datos más importantes en los estados internos del LSTM. Durante la etapa de decodificación, esta representación comprimida funciona como el contexto inicial para volver a armar la secuencia de salida. $\hat{Y}$. Así, el autoencoder LSTM es capaz de aprender a codificar series temporales en un espacio con menos dimensiones, manteniendo tanto las relaciones instantáneas como las dependencias a largo plazo [28], [29].

Los beneficios más destacados de los autoencoders LSTM, en comparación con los autoencoders tradicionales son la habilidad para modelar dependencias temporales complejas, la robustez frente a datos secuenciales que no son estacionarios y la disminución de la información contextual que se pierde durante el proceso de compresión.

No obstante, necesitan más potencia computacional y un ajuste meticuloso de hiperparámetros, incluyendo la tasa de aprendizaje, el tamaño de la ventana temporal y la cantidad de unidades LSTM.

Entrenamiento

La formación de un autoencoder LSTM tiene lugar a través de la propagación hacia adelante, la retropropagación del error y la actualización de parámetros, siguiendo así los mismos principios generales del aprendizaje supervisado o no supervisado que se explicaron en el apartado 2.6.

En este caso el error se mide entre la secuencia de entrada original $X = \{x_1, \dots, x_T\}$ y su reconstrucción $\hat{X} = \{\hat{x}_1, \dots, \hat{x}_T\}$, usando una función de coste del tipo error cuadrático medio (MSE) adaptada a secuencias:

$$C = \frac{1}{T}\sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2 \qquad (2.53)$$

Este coste se propaga hacia atrás en el tiempo mediante el algoritmo de Backpropagation Through Time (BPTT), una extensión del *backpropagation* tradicional que tiene en cuenta la dependencia de los parámetros a lo largo de varios pasos temporales [27], [28].

De forma análoga al proceso explicado en el apartado 2.6, los gradientes de la función de coste respecto a los pesos y sesgos de cada celda LSTM se calculan por medio de la regla de la cadena, pero en este caso se acumulan en el tiempo. Los parámetros $\theta = \{W_f, W_i, W_c, W_o, b_f, b_i, b_c b_o\}$ se actualizan según:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial C}{\partial \theta_t} \qquad (2.54)$$

donde η representa la tasa de aprendizaje.

2.8 Monitorización distribuida

El número de variables monitorizadas ha crecido exponencialmente en los sistemas industriales actuales a causa de la digitalización, la sensorización y el empleo de sistemas ciberfísicos. Si bien esta abundancia de datos posibilita una observación más exacta del proceso, también presenta significativos retos en cuanto a la administración, el tratamiento y el análisis de la información.

En una perspectiva clásica de supervisión centralizada, todos los datos se reúnen en un solo modelo global que tiene la responsabilidad de identificar errores o desviaciones. Sin embargo, cuando hay muchas variables, este método se torna ineficaz o incluso imposible de implementar, debido a que los

modelos tienden a ser más complejos, difíciles de entrenar y propensos a sobreajustarse. Asimismo, es común que los sistemas industriales se estructuren en subsistemas o subprocesos que son parcialmente autónomos; en estos, la relación entre las variables puede ser local y no necesariamente global [31].

La monitorización distribuida surge para enfrentar estas limitaciones; su principio básico es dividir el conjunto total de variables en subconjuntos o bloques que representen de manera coherente diferentes secciones del proceso. Cada bloque es monitorizado de forma individual a través de modelos locales (como los PCA o autoencoders locales), y después sus resultados se combinan utilizando un procedimiento de fusión estadística o probabilística que posibilita la obtención de una perspectiva global del sistema [32].

Esta perspectiva disminuye la carga computacional, facilita una detección de fallos más localizada y mejora la interpretación de los resultados al identificar cuál componente o bloque del sistema es responsable de una potencial anomalía.

La metodología utilizada para llevar a cabo esta división y la estrategia que se empleó para fusionar los resultados de cada bloque de manera coherente, utilizando inferencia bayesiana, se exponen en las siguientes secciones.

2.8.1 Subdivisión de variables en bloques mediante el método mínima redundancia máxima relevancia

Para diseñar sistemas de monitorización distribuida, es crucial la selección de características, pues hace posible disminuir la dimensionalidad del conjunto de datos mientras se mantienen las variables más informativas. El mRMR (Minimum Redundancy - Maximum Relevance), el cual fue propuesto por Peng et al. en 2005 [31], es uno de los métodos más empleados. Este escoge un subconjunto de características que tienen una alta relevancia en relación a la variable objetivo y una baja redundancia entre ellas.

La base del método mRMR es la información mutua (MI, por sus siglas en inglés), que calcula el grado de dependencia entre dos variables. Dada una variable de entrada x_i, una variable de salida y, la información mutua es definida como:

$$I(x_i;y) = \int\int p(x_i,y)\log\frac{p(x_i,y)}{p(x_i)p(y)}\,dx_i\,dy \tag{2.55}$$

donde $p(x_i, y)$ es la distribución conjunta de ambas variables y $p(x_i), p(y)$ son las distribuciones marginales. El valor de $I(x_i; y)$ es directamente proporcional a la implicación que tiene la variable x_i sobre la salida y.

El criterio de **máxima relevancia** busca seleccionar las variables más relacionadas con la salida del sistema:

$$maxD, \quad D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; y) \tag{2.56}$$

donde S es el conjunto de variables seleccionadas.

Sin embargo, seleccionar únicamente las variables más relevantes puede generar redundancia si varias de ellas aportan información similar. Por ello, el método introduce el segundo criterio de **mínima redundancia**, definido como:

$$minR, \quad R = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i; x_j) \tag{2.57}$$

El objetivo final de mRMR es **maximizar la relevancia y minimizar la redundancia simultáneamente**, lo que se formula como:

$$max\Phi, \quad \Phi = D - R \tag{2.58}$$

Este balance garantiza que el grupo de variables escogidas recoja la mayor cantidad posible de información pertinente acerca del proceso, evitando incluir variables que estén correlacionadas entre sí.

Después de que se ha calculado la matriz de información mutua entre todas las variables del proceso, es factible clasificarlas en bloques según su dependencia funcional o estadística. Se considera que las variables que tienen la mayor información mutua entre ellas pertenecen al mismo subsistema físico o funcional del proceso; por lo tanto, se agrupan en el mismo bloque.

Así, el método mRMR no solo hace posible la selección de las características más relevantes, sino también la creación de una estructura distribuida natural del sistema. Esta puede utilizarse para construir modelos de vigilancia local autónomos que luego se combinan a través de métodos de inferencia bayesiana, que se explican en la siguiente sección.

2.8.2 Inferencia Bayesiana

Después de hacer la división de las variables en B bloques a través del método mRMR, cada bloque produce sus propios estadísticos de control T_b^2 y Q_b, los cuales posibilitan la monitorización del comportamiento local de cada bloque. No obstante, para calcular una medida del rendimiento total del sistema, se debe integrar la información de todos los bloques de manera coherente. Se utiliza la inferencia bayesiana, un instrumento estadístico que posibilita la actualización de la probabilidad de un evento (como puede ser la ocurrencia de un error) conforme se va adquiriendo nueva evidencia en cada bloque [27], [28], [33].

Esta investigación se basa en la combinación bayesiana de estadísticos locales, lo que da origen a los BIC (Bayesian Inference Combination), que funcionan como indicadores generales de desviación del proceso. Esta estrategia posibilita la integración de los datos de los bloques, teniendo en cuenta su fiabilidad y su aporte a la identificación de fallos, lo que a su vez brinda una reconstrucción más sólida de las estadísticas globales T^2 y Q.

Fundamento teórico

De acuerdo con la regla de probabilidad condicional de Bayes, la probabilidad de fallo del estadístico T^2 en una muestra x_b perteneciente al bloque b se expresa como:

$$P_{T^2}(F|x_b) = \frac{P_{T^2}(x_b|F)P_{T^2}(F)}{P_{T^2}(x_b)} \qquad (2.59)$$

Donde $P_{T^2}(F|x_b)$ es la probabilidad condicional de observar los datos x_b bajo la hipótesis de fallo (F) y $P_{T^2}(F)$ es la probabilidad a priori de fallo. El denominador puede expresarse como la suma de las probabilidades de operación normal o fallo $(N), (F)$:

$$P_{T^2}(x_b) = P_{T^2}(x_b|N)\ P_{T^2}(N) + \ P_{T^2}(x_b|F)P_{T^2}(F) \qquad (2.60)$$

Las probabilidades condicionales pueden modelarse mediante funciones exponenciales que dependen del valor del estadístico $T_b^2(x_b)$ y su umbral de control $T_{b,lim}^2$:

$$P_{T^2}(x_b|N) = e^{-T_b^2(x_b)/T_{b,lim}^2} \qquad (2.61)$$

$$P_{T^2}(x_b|F) = e^{T^2_{b,lim}/-T^2_b(x_b)} \quad (2.62)$$

Estas formulaciones permiten ponderar la probabilidad de que un bloque esté bajo condiciones normales o de fallo, en función del grado de desviación de sus valores $T_b^2(x_b)$ respecto a su umbral de control $T^2_{b,lim}$.

Combinación Bayesiana de Bloques

Después de calcular las probabilidades parciales de cada bloque, los BICs se encargan de combinarlas en un valor global, ponderando cada bloque con base en su probabilidad de fallo:

$$BIC_{T^2} = \sum_{h=1}^{B} \frac{P_{T^2}(x_b|F)P_{T^2}(F|x_b)}{\sum_{b=1}^{B} P_{T^2}(X_b|F)} \quad (2.63)$$

$$BIC_Q = \sum_{h=1}^{B} \frac{P_Q(x_b|F)P_Q(F|x_b)}{\sum_{b=1}^{B} P_Q(X_b|F)} \quad (2.64)$$

Los estadísticos combinados del sistema son los valores BIC_{T^2} y BIC_Q, los cuales pueden ser utilizados para detectar anomalías de la misma manera que las estadísticas tradicionales T_2 y Q.

Se establece el límite de confianza (α) siguiendo el mismo criterio que en los casos individuales, de modo que, si los BICs sobrepasan sus respectivos límites, se considera que hay un posible fallo global en la planta.

Capítulo 3. Proceso Tennessee-Eastman

3.1. Origen y contexto histórico

Uno de los modelos de referencia más empleados en la ingeniería química y control de procesos es el proceso Tennessee-Eastman (TE). En 1993, Downs y Vogel plantearon un problema de control a nivel de planta (plant-wide control problem) que fue concebido como un banco de pruebas estandarizado para cotejar métodos de detección de fallos y estrategias de control industrial [34].

El modelo fue creado a partir de un proceso químico industrial auténtico de la Tennessee Eastman Company, aunque se hicieron cambios intencionados para salvaguardar la propiedad intelectual del procedimiento original. Se llevó a cabo su implementación en FORTRAN, añadiendo las reacciones químicas, los balances de energía y masa, las ecuaciones dinámicas de flujo y los lazos de control descentralizados [35].

La motivación principal para su desarrollo fue disponer de un caso de estudio abierto, reproducible y representativo de una planta industrial compleja, que incluyera múltiples unidades interconectadas, reciclajes y comportamientos no lineales. Desde su publicación, el proceso TE se ha convertido en un estándar de referencia internacional en la evaluación de algoritmos de detección de fallos, control predictivo, aprendizaje automático y monitorización estadística de procesos [36], [37], [38].

3.2. Descripción general del proceso

El proceso TE consta de cinco unidades principales en serie: reactor bifásico con reacción exergónica, condensador, separador vapor-líquido (flash), columna de destilación (stripper) y compresor de reciclaje (Figura 22). Estas unidades están interconectadas por corrientes de alimentación, purga y reciclaje, conformando un sistema fuertemente acoplado y con lazos de control interdependientes [35].

El objetivo del proceso es la producción de dos productos líquidos (G y H) a partir de cuatro reactantes gaseosos (A, C, D y E), en presencia de un componente inerte (B) y la generación de un subproducto (F).

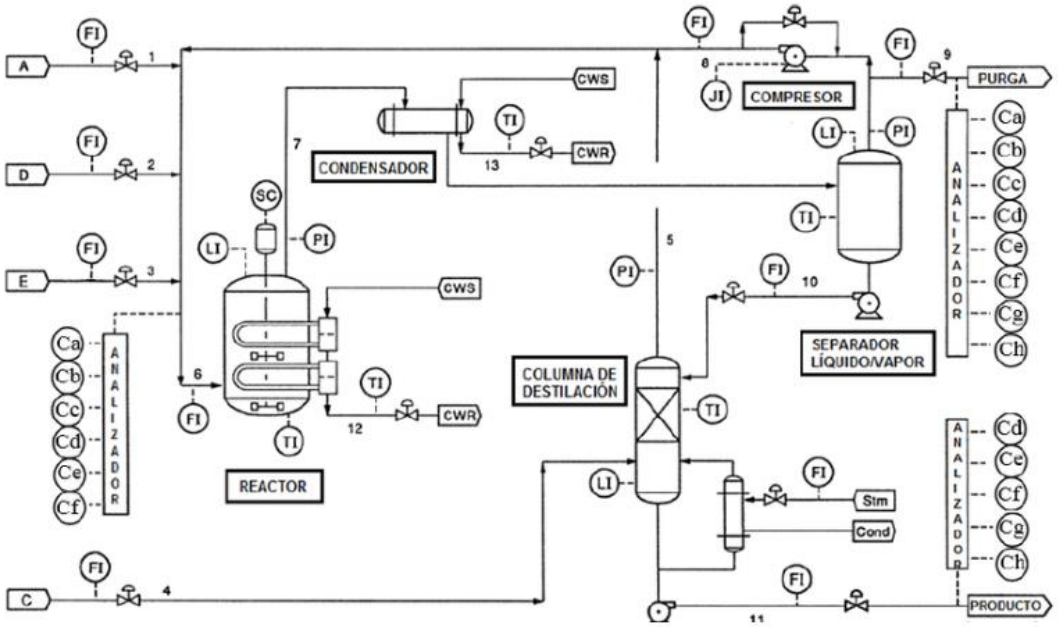


Figura 22. Diagrama ISA del proceso Tennessee-Eastman [35]

Las reacciones químicas principales del proceso se expresan como:

$$A(g) + C(g) + D(g) \rightarrow G(l) \quad (3.1)$$

$$A(g) + C(g) + E(g) \rightarrow H(l) \quad (3.2)$$

$$\frac{1}{3}A(g) + \frac{1}{3}D(g) + \frac{1}{3}E(g) \rightarrow F(l) \quad (3.3)$$

El proceso cuenta con 52 variables medidas y manipuladas, incluyendo presiones, temperaturas, niveles de líquido, fracciones molares, flujos, y aperturas de válvulas. De ellas, 41 son variables de proceso (XMEAs) y 11 variables manipuladas (XMVs) [35], [38], como se ve en la Tabla 1.

Variables de proceso	Variables de control	Variables manipuladas
1 Alimentación A	23 Comp. A de alimentación	42 Flujo de alimentación D
2 Alimentación D	24 Comp. B de alimentación	43 Flujo de alimentación E
3 Alimentación E	25 Comp. C de alimentación	44 Flujo de alimentación A
4 Alimentación Total	26 Comp. D de alimentación	45 Flujo total de alimentación
5 Flujo de Reciclaje	27 Comp. E de alimentación	46 Válvula de reciclaje compresor
6 Caudal de alimentación	28 Comp. F de alimentación	47 Válvula de purga
7 Presión del reactor	29 Comp. A de la purga	48 Flujo de producto separador
8 Nivel del reactor	30 Comp. B de la purga	49 Flujo de producto purgador
9 Temperatura del reactor	31 Comp. C de la purga	50 Válvula de vapor del purgador
10 Caudal de purga	32 Comp. D de la purga	51 Flujo enfriamiento de reactor
11 Temperatura separador	33 Comp. E de la purga	52 Flujo enfriamiento condensador
12 Nivel del separador	34 Comp. F de la purga	
13 Presión del separador	35 Comp. G de la purga	
14 Desbordamiento separador	36 Comp. H de la purga	
15 Nivel del purgador	37 Comp. D del producto	
16 Presión del purgador	38 Comp. E del producto	
17 Desbordamiento purgador	39 Comp. F del producto	
18 Temperatura del purgador	40 Comp. G del producto	
19 Caudal de vapor purgador	41 Comp. H del producto	
20 Trabajo del compresor		
21 Tª de salida del agua		
22 Tª de salida del agua		

Tabla 1. Descripción de las variables del proceso Tennessee-Eastman [37]

3.3. Formulación matemática del modelo

El modelo Tennessee-Eastman está basado en ecuaciones dinámicas no lineales derivadas de los balances de masa y energía de cada componente y unidad de proceso. A continuación, se resumen sus ecuaciones fundamentales.

a) Balance general de materia

Para cada componente *i* en una unidad *k*:

$$\frac{dNi,k}{dt} = \sum_{i} y_{i,j} F_j^{in} \; - m \sum_{m} yi, mF_m^{out} \; + r \sum_{r} v_{i,r} \; R_r \qquad (3.4)$$

donde:

- Ni,k: cantidad de moles del componente *i* en la unidad *k*
- $y_{i,j}$: fracción molar del componente *i* en la corriente *j*
- F_j^{in} y F_m^{out}: caudales molares de entrada y salida
- $\nu_{i,r}$: coeficiente estequiométrico de la reacción *r*
- R_r: velocidad de reacción del proceso

b) Cinética de reacción

Las velocidades de reacción se describen mediante cinética de Arrhenius de la forma general:

$$R_r = \prod_i C_i^{\alpha_{i,r}} \exp\left(-\frac{E_{a,r}}{RT}\right) \tag{3.5}$$

donde $k_r$es la constante pre-exponencial, $C_i$la concentración del componente i, $E_{a,r}$la energía de activación y T la temperatura del reactor.

c) Relaciones de equilibrio termodinámico

En el separador y el condensador se asume equilibrio vapor-líquido:

$$y_i P = x_i P_i^{sat}(T) \tag{3.6}$$

$$\sum_i y_i = 1 \quad ; \quad \sum_i x_i = 1 \tag{3.7}$$

donde *yi* y *xi* son las fracciones molares en las fases vapor y líquida respectivamente, *P* la presión total y $P_i^{sat}(T)$ la presión de saturación del componente *i* a la temperatura *T*.

3.4. Condiciones de operación y control

El proceso opera bajo condiciones nominales definidas para cada corriente y unidad de proceso. Los lazos de control regulan variables críticas como la temperatura y presión del reactor, el nivel del separador o la fracción molar de productos mediante las 11 variables manipuladas (XMVs).

ID de Fallo	Descripción	Tipo	Magnitud
IDV1	Relación de alimentación A/C comp. B constante	Escalón	203%
IDV2	Composición B relación A/C constante	Escalón	105%
IDV3	Temperatura de alimentación D	Escalón	5%
IDV4	Temperatura de entrada del agua al reactor	Escalón	9%
IDV5	Temperatura de entrada del agua al condensador	Escalón	15%
IDV6	Pérdida de alimentación A	Escalón	342%
IDV7	Pérdida de presión del cabezal C	Escalón	25%
IDV8	Composición de alimentación A B C	Aleatorio	736%
IDV9	Temperatura de alimentación D	Aleatorio	8%
IDV10	Temperatura de alimentación C	Aleatorio	112%
IDV11	Temperatura de entrada del agua al reactor	Aleatorio	567%
IDV12	Temperatura de entrada del agua al condensador	Aleatorio	8%
IDV13	Cinética de reacción	Desviación	16%
IDV14	Válvula de agua al reactor	Bloqueo	1285%
IDV15	Válvula de agua al condensador	Bloqueo	5%
IDV16	Desconocido	Aleatorio	78%
IDV17	Desconocido	Aleatorio	557%
IDV18	Desconocido	Escalón	57%
IDV19	Desconocido	Aleatorio	73%
IDV20	Desconocido	Aleatorio	310%
IDV21	Desconocido	Desconocido	? %

Tabla 2. Fallos definidos en el proceso T-E [40]

El conjunto de 21 fallos o perturbaciones (IDVs) definidos en el modelo (Tabla 2) permiten estudiar condiciones anómalas como:

- Cambios en la temperatura o composición de alimentación.
- Fallos en válvulas o restricciones de flujo.
- Alteraciones en la cinética de reacción.
- Perturbaciones aleatorias o desviaciones lentas.

Estos modos de fallo son ampliamente utilizados como casos de prueba en el desarrollo y validación de métodos de detección y diagnóstico de fallos [32], [36].

3.5. Aplicaciones y relevancia

El proceso Tennessee-Eastman es un estándar de referencia internacional en la investigación y desarrollo de metodologías de control y diagnóstico de procesos. Sus principales aplicaciones incluyen [37], [38]:

- Evaluación de métodos de detección e identificación de fallos (PCA, PLS, ICA, redes neuronales, autoencoders, etc.).
- Análisis de la estabilidad y robustez de sistemas de control planta-amplia.
- Desarrollo de técnicas de monitoreo estadístico y control predictivo.
- Entrenamiento y validación de modelos de inteligencia artificial para diagnóstico industrial.

Las versiones modernas del simulador y los conjuntos de datos asociados están disponibles públicamente a través de los repositorios del MIT Braatz Group [39] y del Harvard Dataverse [40], los cuales incluyen tanto datos de funcionamiento normal como series con fallos simulados.

3.6. Limitaciones del modelo

Pese a su gran utilidad, el modelo Tennessee-Eastman presenta ciertas limitaciones [35], [36]:

- No incorpora todas las no linealidades y retardos temporales presentes en plantas reales.
- Algunos parámetros cinéticos y termodinámicos fueron modificados para proteger la propiedad industrial.
- Las condiciones de operación son idealizadas y pueden diferir de las industriales.
- No contempla estrategias de control centralizado u optimización económica.

Aun así, su adopción generalizada lo consolida como un caso de referencia esencial en ingeniería química, control de procesos e inteligencia artificial aplicada.

Capítulo 4. Aplicaciones y evaluación de métodos

4.1 Introducción

Los datos del procedimiento Tennessee-Eastman (TE) se utilizan en este capítulo para poner en práctica los conceptos teóricos que se han desarrollado anteriormente. El propósito es analizar la habilidad de diversas metodologías de monitorización para identificar fallos en condiciones similares.

Las técnicas consideradas incluyen:

1. **PCA (Análisis de Componentes Principales)**: referencia clásica en la reducción de dimensionalidad y control estadístico multivariante.
2. **Autoencoder**: red neuronal entrenada de manera no supervisada para aprender correlaciones no lineales.
3. **Autoencoder Recurrente (RAE)**: adaptación que incorpora dinámica temporal en el aprendizaje mediante células LSTM.
4. **RAE distribuido**: extensión que divide las variables en bloques y fusiona la información mediante inferencia bayesiana.

Todas las metodologías se entrenan solamente con datos de operación normal (Tabla 3) y se contrastan con los casos de fallo establecidos en el proceso Tennessee-Eastman (Tabla 1). La capacidad de cada método para detectar anomalías de manera fiable y rápida se mide con los estadísticos T^2 y Q, lo que permite calcular el rendimiento.

Antes de abordar la implementación, haremos una breve mención de las librerías más importantes que hemos empleado en los programas, todos ellos desarrollados con Python 3.11:

- Numpy para el cálculo vectorial, matricial y tensorial a lo largo de todo el trabajo.
  - Versión: 1.26.4.

- Pandas para cargar los datos de la planta y para la división en bloques de la 4.5.
  - Versión: 2.1.4.
- Matplotlib para el dibujo de todos los gráficos de control y la visualización de los resultados.
  - Versión: 3.8.0.
- Tensorflow encargado de la computación numérica optimizada para el aprendizaje profundo, es el encargado de administrar los recursos del

equipo empleado y de ejecutar los modelos de los apartados 4.3, 4.4 y 4.5.
    - Versión: 2.16.
- Keras, incluido dentro de Tensorflow, se emplea como interfaz de alto nivel para definir las arquitecturas de las redes neuronales en las secciones mencionadas anteriormente.
    - Versión: 3.3.3.

- Pickle para guardar y serializar objetos de Python en un archivo binario y luego poder desempaquetarlos más adelante, empleado en el apartado 4.5 a la hora de empaquetar los datos del autoencoder distribuido en un único archivo
    - Versión: 3.8

	VARIABLE							
MUESTRA	1	2	…	14	15	…	51	52
0	0.24889	3702.3	…	25.184	50.201	…	41.384	18.905
1	0.24904	3666.2	…	26.589	49.824	…	41.658	18.976
2	0.25034	3673.3	…	24.494	48.957	…	41.721	16.562
3	0.25109	3657.8	…	27.367	49.708	…	40.836	20.094
4	0.24563	3698	…	22.341	49.662	…	41.727	18.33
5	0.24759	3687.4	…	24.433	51.704	…	40.922	19.532
6	0.24689	3619.7	…	25.761	48.912	…	40.562	21.019
…	…	…	…	…	…	…	…	…
956	0.23352	3625.4	…	24.549	50.322	…	40.971	15.621
957	0.2344	3660.3	…	24.501	48.908	…	41.891	21.744
958	0.23611	3645	…	25.059	47.456	…	39.813	18.826
959	0.23729	3666.8	…	23.602	47.656	…	40.5	18.353

Tabla 3. Datos del funcionamiento normal de la planta

4.2 PCA

Se ha puesto en práctica el método de Análisis de Componentes Principales (PCA) como una herramienta para la detección de errores y la reducción de dimensiones. Su operación se basa en determinar las direcciones de mayor variabilidad de los datos del funcionamiento normal del proceso y representarlas en un espacio más pequeño, manteniendo la información más importante.

Es factible, a partir de este modelo reducido, cotejar datos nuevos con el comportamiento normal aprendido y así identificar potenciales anomalías o desvíos que revelen un fallo en el sistema.

4.2.1 Entrenamiento

En la fase de entrenamiento, se ha utilizado el conjunto de datos correspondiente al funcionamiento normal del proceso, como se observa en la Tabla 3. Primero, las variables se normalizan restando la media y dividiendo por la desviación típica, (2.9), de modo que todas contribuyan de forma equilibrada al modelo.

A continuación, se calcula la matriz de covarianza (2.11) y se obtiene su descomposición en valores singulares (SVD), a partir de la cual se seleccionan los autovectores asociados a los mayores autovalores, que representan las direcciones de máxima varianza del sistema (2.12) .

El número de componentes principales se establece automáticamente en base al porcentaje acumulado de variabilidad, en este caso el 90%. Así, se mantiene la mayoría de los datos útiles y se disminuye la complejidad del problema.

Con los componentes seleccionados se obtiene, la matriz de *loadings* P, la matriz diagonal S de varianzas principales (Figura 8) y los estadísticos de control T^2 *y* Q (o SPE, *Squared Prediction Error*), que cuantifican respectivamente la variabilidad dentro y fuera del subespacio principal como se observa en la Figura 23.

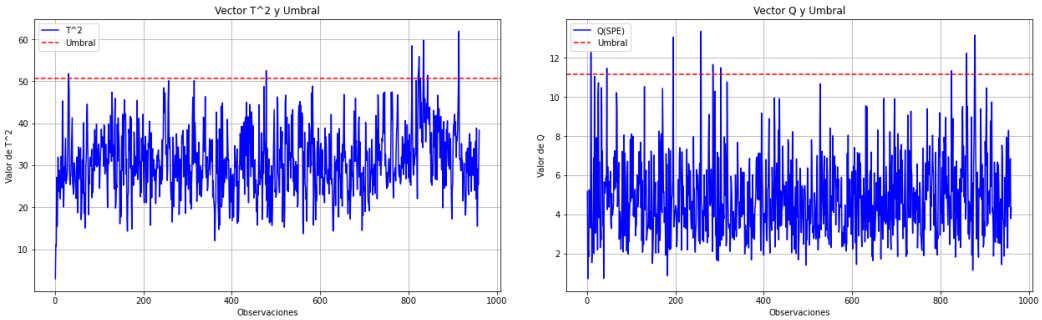


Figura 23. Estadísticos T^2 y Q en PCA. Funcionamiento normal

Finalmente, se calculan los umbrales de control a partir del percentil 99 % de cada estadístico, almacenándose junto con los parámetros del modelo para su posterior uso en la fase de detección.

4.2.2 Detección de Fallos

Después de que el modelo PCA ha sido entrenado, se pasa a la etapa de detección, en la cual los parámetros adquiridos se aplican a un nuevo conjunto de datos que representa una circunstancia del proceso que puede ser anómala.

Primero, se cargan los datos de normalización del entrenamiento y, además, las medias, desviaciones típicas, vectores de carga y límites de control que se han guardado con anterioridad. Los registros del sistema se escalan con las estadísticas del modelo, asegurando que sean coherentes con las condiciones de entrenamiento.

Los estadísticos de control T^2 y Q se determinan para cada observación, del mismo modo que en el entrenamiento, con los mismos datos de normalización.

Ambos indicadores se comparan con sus respectivos umbrales de referencia, los cuales son el percentil 99 % de los datos normales. Cuando el valor de alguno de estos estadísticos supera el umbral ya mencionado, se entiende que la muestra está fuera de control.

El estadístico T^2 evalúa la variabilidad de las observaciones dentro del subespacio principal, permitiendo identificar desviaciones respecto al comportamiento normal de las combinaciones lineales más significativas.

Por su parte, el estadístico Q mide el residuo o error de reconstrucción fuera del subespacio principal, reflejando anomalías que no pueden explicarse mediante las componentes seleccionadas.

A partir de ahora, vamos a representar gráficamente los resultados obtenidos para los datos de fallo 6, 12 y 15, el resto lo representaremos mediante tablas. En concreto para estos fallos los estadísticos T^2 y Q se representan en las Figuras 24, 25 y 26 respectivamente:

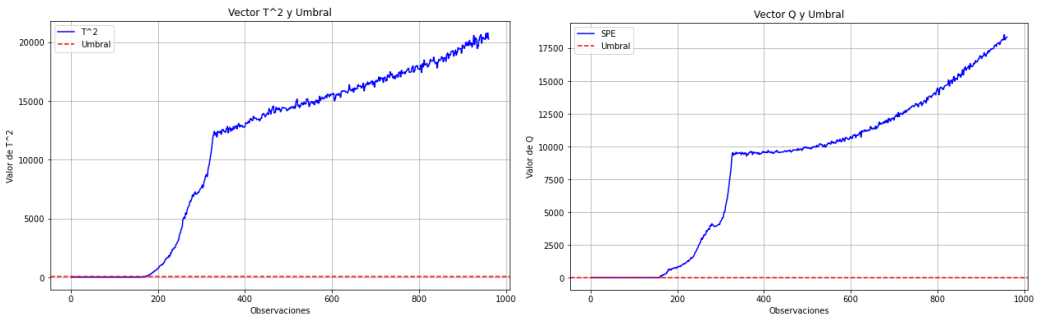


Figura 24. Detección del fallo IDV6 mediante PCA. Estadísticos T^2 y Q

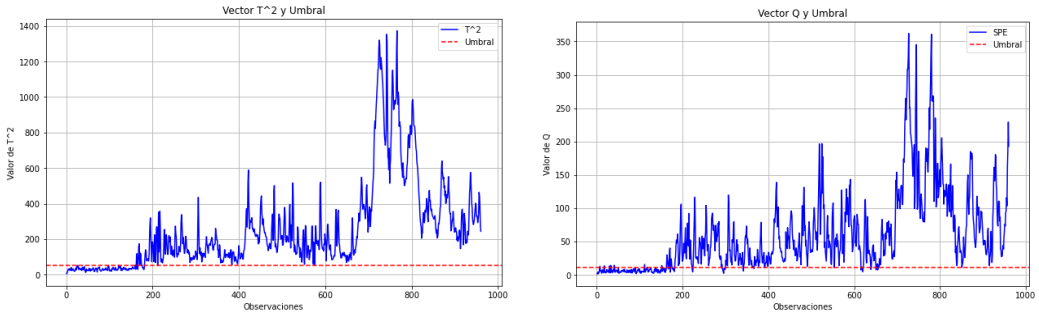


Figura 25. Detección del fallo IDV12 mediante PCA. Estadísticos T^2 y Q

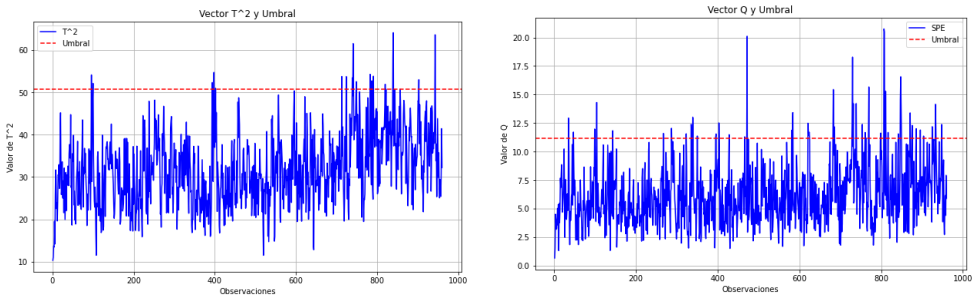


Figura 26. Detección del fallo IDV15 mediante PCA. Estadísticos T^2 y Q

Luego, se lleva a cabo un análisis de la cantidad de alarmas falsas y verdaderas para ambos estadísticos, diferenciando el área de funcionamiento normal del área donde se introduce un error. Además, se define un criterio de fallo sostenido: si una serie ininterrumpida de observaciones (como diez) excede el umbral definido para un comportamiento normal, se establece el instante exacto en que el sistema deja de operar con normalidad, como se observa en la Tabla 4.

Finalmente, se realiza un diagnóstico del motivo del fallo a través del análisis de los residuos sólo en el instante en que se detecta el fallo. Estos se determinan proyectando las observaciones en el espacio ortogonal al subespacio principal y calculando la contribución de cada variable al error total como se ve en la Figura 27.

Lo que permite determinar las variables con mayor impacto en la desviación detectada y, por lo tanto, los posibles motivos del error en el procedimiento. Cabe destacar que esta contribución solamente es calculable si el modelo es capaz de detectar los fallos.

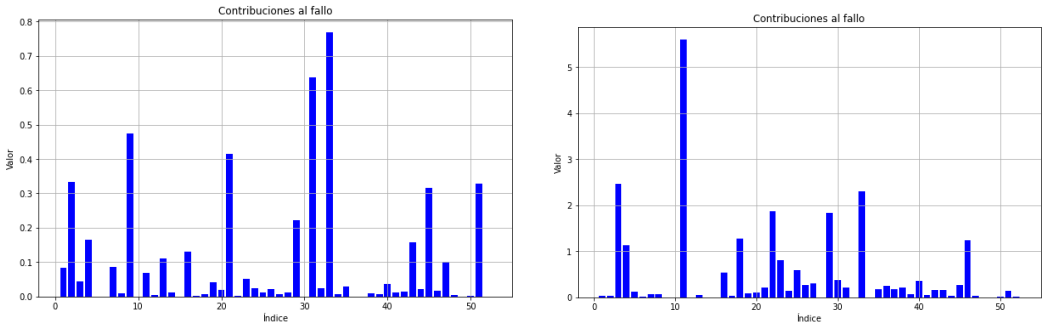


Figura 27. Contribuciones de cada variable. Fallos IDV6 y 12 respectivamente

4.2.3 Resultados PCA

En base a los resultados, observamos que el PCA es capaz de identificar errores en la mayoría de los IDs de fallo (Tabla 4), aunque presenta una media de alarmas detectadas que es relativamente baja (Tabla 5).

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas (%)**	**t_Fallo(obs)**	**FalsasA (%)**	**Alarmas (%)**	**t_Fallo (obs)**
1	1,25	99,25	166	3,75	99,75	162
2	1,875	98,63	171	4,375	98,63	174
3	1,25	1,88	**No Detect.**	2,5	7,50	**No Detect.**
4	0,625	41,88	664	2,5	100,00	160
5	0,625	25,75	170	2,5	33,88	160
6	0,625	98,88	169	2,5	100,00	160
7	0	100,00	160	2,5	100,00	160
8	0	97,25	182	4,375	96,88	177
9	3,125	3,38	**No Detect.**	5	4,50	**No Detect.**
10	1,875	30,75	263	0	46,00	207
11	0	51,25	210	5	69,38	166
12	1,875	98,63	181	6,25	95,13	182
13	1,875	94,63	205	1,25	95,13	200
14	0,625	99,50	160	5	99,88	161
15	1,25	2,50	**No Detect.**	3,125	6,13	**No Detect.**
16	3,75	15,13	470	3,125	43,50	353
17	0,625	78,63	188	3,125	95,63	181
18	1,25	89,38	252	5,625	90,13	243
19	0,625	12,25	**No Detect.**	4,375	21,88	**No Detect.**
20	2,5	31,25	246	3,75	55,88	244
21	1,25	41,38	664	3,75	50,13	409

Tabla 4. Detección de fallos mediante PCA

Promedios PCA			
	Falsas Alarmas (%)	Alarmas (%)	Tiempo de detección (observaciones)
T^2	1,28	57,72	265,94
Q	3,54	67,14	205,82

Tabla 5. Resumen estadístico PCA

Total Fallos PCA	
T^2	17
Q	17

Tabla 6. Fallos detectados con PCA

Por otro lado, se ve que el método no es capaz de detectar 4 fallos (los fallos 3, 9, 15 y 19) ni con la estadística T^2 ni con la Q (Tablas 5 y 6).

En las secciones que siguen, utilizaremos técnicas de aprendizaje profundo con el objetivo de lograr una detección más precisa, dado que pueden aprender patrones de mayor complejidad que los que pertenecen al PCA.

4.3 Autoencoder

La reducción de la dimensionalidad y la identificación de anomalías se llevan a cabo utilizando el método del autoencoder denso, que es un enfoque no lineal. El autoencoder, en contraste con el PCA, que se fundamenta en combinaciones lineales de las variables, emplea una red neuronal con múltiples capas densas conectadas simétricamente para adquirir una representación comprimida de los datos de operación normal.

Durante el proceso de entrenamiento, la red se ajusta para reconstruir la entrada a partir de su versión reducida, de modo que la información esencial del sistema se concentre en las capas internas.

Esta habilidad de reconstrucción posibilita, más adelante, determinar desviaciones importantes entre la señal original y la que ha sido reconstruida; esto se convierte en el fundamento para detectar fallos.

4.3.1 Entrenamiento

El pretratamiento de los datos se realiza de manera idéntica en todos los métodos, pero esta vez partimos de una matriz de datos mucho más grande, de 250.000 observaciones exactamente, tanto para este autoencoder, como

para el resto, esto es asi porque para obtener un buen entrenamiento de la red se necesita una gran cantidad de datos.

La arquitectura del autoencoder implementado está compuesta por una red simétrica que tiene tres capas ocultas en el codificador y tres en el decodificador siendo la representación latente (h) compartida en encoder y decoder como vimos en la Figura 18, las cuales están organizadas con las dimensiones mostradas en la Tabla 7.

Capa	Nombre de capa	Tipo de capa	Dimensión	Función de activación	Tipo
0	Input	Entrada	52	ReLU	Entrada
1	encoded1	Dense	48	ReLU	Encoder
2	encoded2	Dense	35	LeakyReLU (α=0.1)	Encoder
3	hidden (Dense)	Dense	24	ReLU	Encoder (hidden)
5	decoded2	Dense	35	ReLU	Decoder
6	decoded1	Dense	48	Sigmoid	Decoder
7	output	Salida	52	-	Salida

Tabla 7. Configuración Autoencoder

Las capas utilizan funciones de activación ReLU para las capas de codificación lo que permite modelar relaciones no lineales complejas entre las variables y LeakyReLU para la capa oculta con el objetivo de mitigar el problema del desvanecimiento del gradiente mencionado anteriormente y finalmente sigmoide en la salida.

La finalidad del modelo es la de reducir el error cuadrático medio (MSE) (2.40) entre la entrada y la salida reconstruida, para ello, se entrena el modelo utilizando Adam como optimizador, ya mencionado en el apartado 2.6.4.1 Adam (Adaptive Moment Estimation).

En el entrenamiento, se deja un 20 % de los datos para validación. Con un tamaño de lote de 64 muestras, el aprendizaje tiene lugar en un total de 30 épocas. El resultado de entrenamiento, se observa en la Figura 28, y se ve como con la arquitectura de red seleccionada se consigue una convergencia rápida y sin sobreajuste.

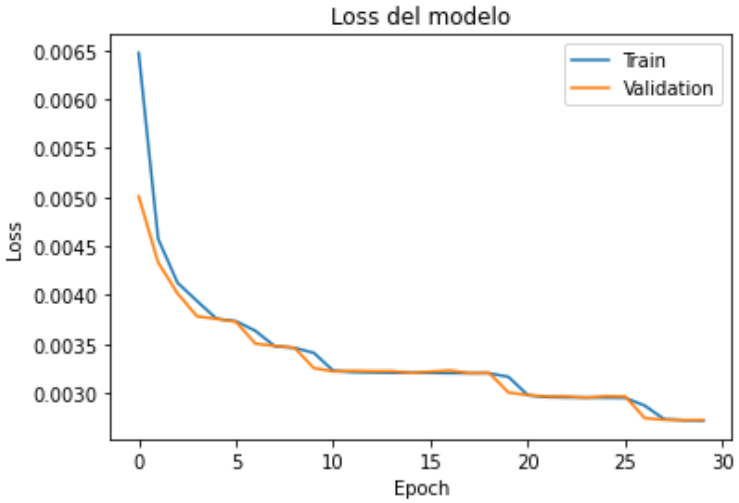


Figura 28. Curvas de pérdida en entrenamiento y validación del autoencoder

Después de que el entrenamiento ha concluido, se obtiene el codificador, que convierte las observaciones iniciales en un espacio de dimensiones reducidas h. Entonces, se calcula la matriz de covarianza de las representaciones comprimidas en este nuevo espacio.

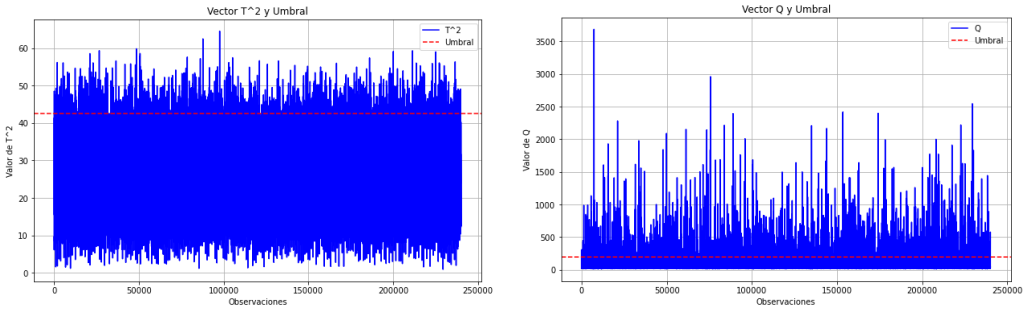


Figura 29. Estadísticos T^2 y Q en Autoencoder. Funcionamiento normal

En función de ella, se establece el estadístico T^2, que analiza la variabilidad en el subespacio codificado, y el estadístico Q, que calcula el residuo de reconstrucción (diferencia entre entrada y salida) como se ve en la Figura 29, pero esta vez ambos estadísticos han sido calculados como:

$$T_i^2 = (h_i - \mu_h) \cdot \ cov(h) \cdot (h_i - \mu_h)^t \qquad (4.1)$$

$$Q_i = (r_i - \mu_r) \cdot (r - \mu_r)^t \qquad (4.2)$$

Siendo h_i la representación latente de la red, equivalente a los autovectores obtenidos en PCA, r_i el residuo calculado como $X_n - \hat{X}$ siendo X_n la matriz de

datos normalizada correspondiente a la entrada de la red y $\hat{X}$ la reconstrucción del autoencoder. μ_h y μ_r sus medias, respectivamente y $cov()$ la covarianza.

Los límites que determinan el funcionamiento normal del sistema se establecen utilizando los percentiles 99 % del conjunto de entrenamiento, con lo cual se obtienen los umbrales de control para ambos estadísticos.

Por último, se almacenan en archivos los modelos entrenados (autoencoder y encoder) junto con los parámetros estadísticos y umbrales, que se utilizarán en la fase de detección.

4.3.2 Detección de fallos

En la etapa de detección, se emplean los modelos que han sido entrenados antes con un nuevo conjunto de datos con fallos.

Se normalizan las observaciones empleando los parámetros del conjunto de entrenamiento y se proyectan en el espacio latente a través del codificador, lo que produce sus reconstrucciones pertinentes con el autoencoder.

Con estos resultados, se vuelven a calcular los estadísticos de control T^2 y Q para cada observación, y fallo utilizando las mismas pautas que durante el entrenamiento. Ambos se comparan con sus límites respectivos para establecer si el proceso está en un estado normal o descontrolado. Mostramos los resultados de estas estadísticas junto con sus umbrales para los fallos 6, 12 y 15 en las figuras 30, 31, y 32 respectivamente. Destacando los instantes en los que las observaciones superan los límites establecidos.

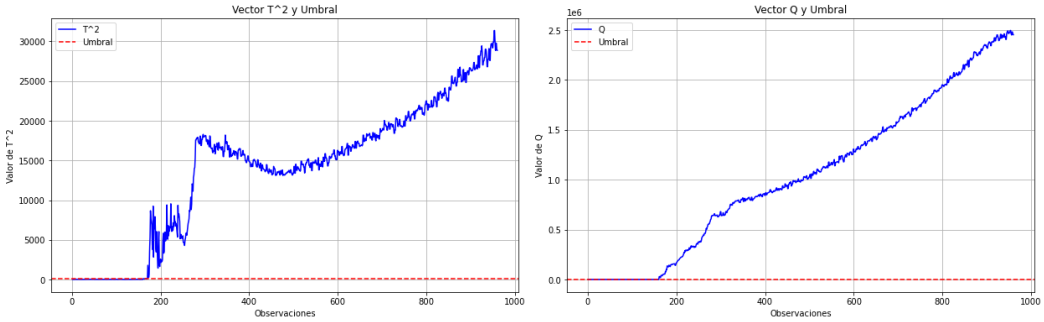


Figura 30. Detección del fallo IDV6 con Autoencoder. Estadísticos T^2 y Q

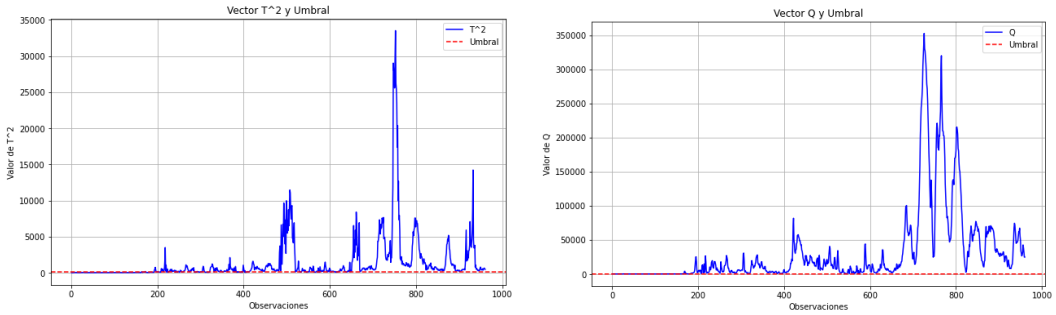


Figura 31. Detección del fallo IDV12 con Autoencoder. Estadísticos T^2 y Q

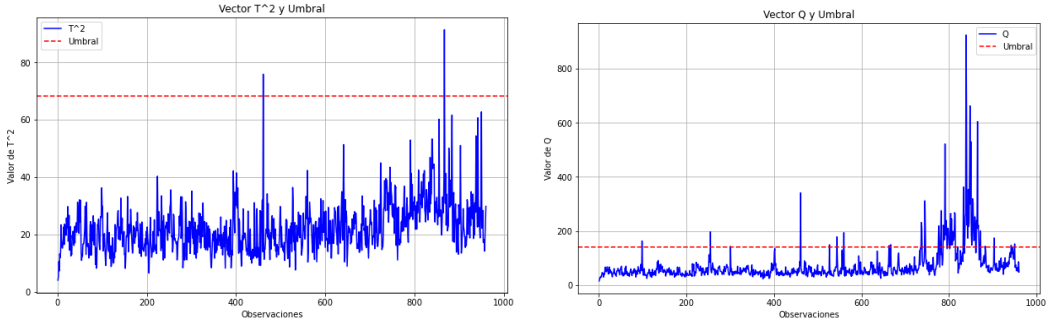


Figura 32. Detección del fallo IDV15 con Autoencoder. Estadísticos T^2 y Q

Con base en estos gráficos, se cuentan las alarmas falsas (que corresponden a superaciones del umbral en la zona de funcionamiento normal) y las alarmas verdaderas (que son las que ocurren al sobrepasar el umbral en la zona de fallo).

Asimismo, se aplica un criterio de error sostenido, para que, si una serie continua de muestras (diez, por ejemplo) excede el umbral, se determine el momento exacto en que el sistema falla, obteniendo para los 21 fallos los estadísticos de la Tabla 8 y los estadísticos promedio de la Tabla 9 y la Tabla 10.

Por último, para determinar la causa del fallo, se examina el vector de residuos que corresponde con el momento en que fue detectado.

Se representa en un gráfico de barras la contribución individual de cada variable como se observa en la Figura 33, que se obtiene del cuadrado de los residuos. Esto posibilita observar qué variables tienen mayor impacto en la anomalía detectada, lo cual ayuda a identificar la causa del error. En este caso, la Figura 33 nos indica que las variables 1 y 44 son las variables que provocan el fallo 6, así como la variable 22 es la que provocó el fallo 12. El fallo 15 parece ser provocado por demasiadas variables, por lo que no nos da mucha información.

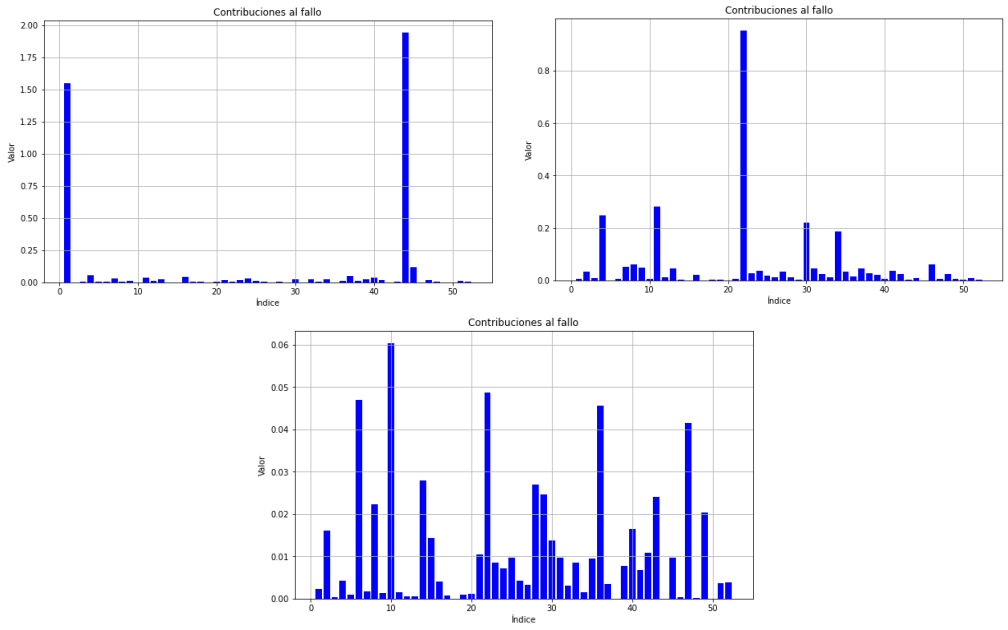


Figura 33. Contribuciones al fallo IDV6, IDV12 e IDV15 en Autoencoder

4.3.3 Resultados

Como hicimos antes, comentaremos algunos gráficos de control significativos y pasaremos a ver los estadísticos de cada fallo y los generales.

El fallo 6 (Figura 30) se detecta de forma similar al método anterior, los tiempos de observación son similares, sin embargo, las alarmas que detecta el autoencoder son mucho mayores.

El fallo 12 (Figura 31) En este caso el estadístico T^2 de PCA detecta un poco mas rápido el fallo, sin embargo, es el estadístico Q del Autoencoder quien ha sido capaz de detectar antes de forma general el fallo.

Por último, el fallo 15 (Figura 32) ahora pasa a ser detectado por el estadístico Q lo que, junto al resultado del fallo 12, denota como el autoencoder es capaz de almacenar más información que el método estadístico de PCA

De forma general observamos que el comportamiento del autoencoder es superior al de PCA, si bien es cierto que este mantiene una mayor homogeneidad en la detección de fallos con relación a sus estadísticos, el autoencoder ha demostrado que el estadístico Q es superior a la hora de detectar fallos, incurriendo en una mayor detección total de fallos. Por otro lado, destacar que se han reducido drásticamente el índice de falsas alarmas, especialmente en el estadístico T^2.

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas(%)**	**t_Fallo(obs)**	**FalsasA(%)**	**Alarmas(%)**	**t_Fallo(obs)**
1	0	99,25	166	1,25	99,50	164
2	0	98,25	174	1,25	98,50	172
3	0	1,00	**No Detect.**	5,625	6,13	**No Detect.**
4	0,625	1,88	**No Detect.**	2,5	70,50	165
5	0,625	22,63	172	2,5	100,00	160
6	0	99,63	168	0,625	100,00	160
7	0	84,00	160	0,625	99,88	160
8	0,625	97,00	181	0,625	98,00	179
9	1,25	1,38	**No Detect.**	11,875	6,88	120
10	1,25	18,38	261	1,25	77,00	184
11	0	11,75	**No Detect.**	0,625	55,25	255
12	0	95,13	191	4,375	99,00	162
13	0	93,63	206	0	95,13	203
14	0	99,75	161	0,625	99,88	161
15	0	0,25	**No Detect.**	0,625	10,13	802
16	2,5	3,63	**No Detect.**	25	79,13	170
17	0,625	72,25	186	0,625	91,25	183
18	0	89,13	250	1,875	89,75	244
19	0	2,00	**No Detect.**	0	56,38	170
20	0	25,00	252	0	71,25	231
21	0	20,38	848	7,5	38,38	674

Tabla 8. Detección de fallos mediante autoencoder

Promedios Autoencoder			
	Falsas Alarmas (%)	**Alarmas (%)**	**Tiempo de detección (observaciones)**
T^2	0,36	49,35	241,14
Q	3,30	73,42	235,95

Tabla 9. Resumen estadístico Autoencoder

Total Fallos Autoencoder	
T^2	14
Q	20

Tabla 10. Fallos detectados con Autoencoder

Las alarmas siguen la misma dinámica que los estadísticos, el autoencoder detecta mayor porcentaje de forma absoluta con su estadístico Q, pero el T^2 de PCA es superior detectando fallos.

4.4 Autoencoder Recurrente (RAE)

El modelo que se ha creado está fundamentado en una arquitectura de autoencoder secuencial que se pone en práctica a través de redes LSTM (Long Short-Term Memory). Estas redes son particularmente apropiadas para el tratamiento de series temporales y señales que dependen del tiempo, porque posibilitan la captura de relaciones dinámicas a través de diversas escalas temporales.

El sistema está compuesto por dos segmentos fundamentales: el codificador (encoder) y el decodificador (decoder), que están conectados mediante un cuello de botella o capa latente, la cual funciona como una representación comprimida de la serie de entrada. El propósito del modelo, en resumen, es adquirir una representación eficaz de los datos temporales y usarla para reconstruir la secuencia original.

4.4.1 Entrenamiento

Se emplean otra vez los datos del proceso en condiciones normales para entrenar el modelo RAE. Se quitan las columnas iniciales que no aportan información y se normalizan todas las variables entre 0 y 1 utilizando sus valores máximos y mínimos.

Después, las observaciones se reestructuran en ventanas temporales deslizantes de cinco muestras, permitiendo que el modelo capte dependencias entre los valores presentes y los pasados de las variables a corto y medio plazo.

El codificador se compone de dos capas LSTM apiladas secuencialmente. La capa inicial toma como entrada una secuencia de múltiples dimensiones y produce una nueva secuencia de salida donde cada paso temporal se convierte en un vector de activación con una dimensión constante. Esta capa utiliza una función de activación no lineal conocida como Scaled Exponential Linear Unit (SELU), que permite la auto-normalización de las activaciones y optimiza la estabilidad en el proceso de entrenamiento.

La segunda capa LSTM recibe la secuencia generada por la primera y la sintetiza en un solo vector de menor tamaño, extrayendo los datos más significativos de toda la serie temporal. En esta etapa, el modelo deja de devolver una secuencia y brinda en su lugar una representación latente fija, que se considera el vector codificado del autoencoder o capa oculta.

El vector creado por el codificador simboliza el estado comprimido de la secuencia original. Este vector incluye la información crucial para reconstruir la señal de entrada con el mínimo de pérdida posible, funcionando como un espacio latente de características del modelo.

Para el decodificador, primeramente, se emplea una capa de repetición (Repeat Vector) que reproduce el vector latente el número de veces que la secuencia de entrada tiene pasos temporales. Este proceso produce una secuencia inicial artificial que será la entrada del decodificador en sí.

Se añaden dos capas LSTM más a continuación. La primera conserva la misma dimensión que el vector latente, pero la segunda aumenta el número de unidades de salida para acercarse gradualmente a la complejidad de la secuencia inicial. Las dos capas utilizan, además, la activación SELU y devuelven secuencias completas, lo que asegura que cada momento temporal se reconstituyera de forma coherente a través del tiempo.

Por último, se añade una capa densa distribuida en el tiempo (TimeDistributed Dense Layer), la cual se aplica de manera independiente a cada paso de la secuencia que ha sido reconstruida. Esta capa envía los vectores de activación del decodificador hacia el espacio de salida, creando una secuencia con la misma dimensionalidad que la entrada inicial. Su activación es lineal porque el propósito es reducir al mínimo el error de reconstrucción sin imponer restricciones extra a los valores de salida. El resumen de la red con todas sus capas y dimensiones se encuentra en la Tabla 11.

Capa	Nombre de capa	Tipo de capa	Dimensión	Función de activación	Topi
0	Input	Entrada	(5, 52)	—	Entrada
1	encoded1	LSTM	(5, 48)	SELU	Encoder
2	encoded2	LSTM	24	SELU	Encoder (hidden)
3	repeat_vector	RepeatVector	(5, 24)	—	Decoder
4	decoded1	LSTM	(5, 24)	SELU	Decoder
5	decoded2	LSTM	(5, 48)	SELU	Decoder
6	output	Dense	52	Lineal	Salida

Tabla 11. Configuración del Autoencoder LSTM

La red se configura de manera que la salida del decodificador tenga la misma forma que la secuencia de entrada.

Durante el entrenamiento, el modelo se optimiza mediante una función de pérdida de tipo error cuadrático medio (*Mean Squared Error*, MSE), que cuantifica la diferencia entre la secuencia original y su reconstrucción, utilizando el optimizador Adam con una tasa de aprendizaje de 0.001.

Se emplea un 20 % de los datos como conjunto de validación para controlar el sobreajuste, y el proceso se ejecuta durante 4 épocas con un tamaño de lote de 32 muestras.

De esta forma, el autoencoder aprende a representar y reproducir las dinámicas temporales de los datos, capturando las dependencias secuenciales más relevantes dentro de un espacio de representación comprimido.

Al finalizar el entrenamiento, se evalúa la convergencia del modelo mediante las curvas de pérdida de entrenamiento y validación, tal y como se ve en la Figura 34.

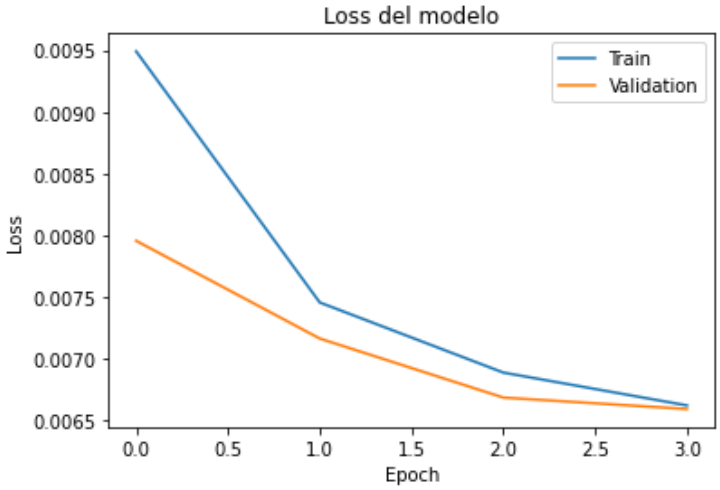


Figura 34. Curva de pérdidas del modelo durante el entrenamiento.

A continuación, se genera el modelo encoder, que transforma cada secuencia de entrada en un vector reducido que representa el estado dinámico del sistema.

Sobre este espacio latente se calculan las estadísticas de control Hotelling's T^2 y Q, que permiten cuantificar, respectivamente, la variabilidad explicada y el error de reconstrucción de cada secuencia. Estas estadísticas calculadas para el comportamiento normal de la planta se pueden ver en la Figura 35.

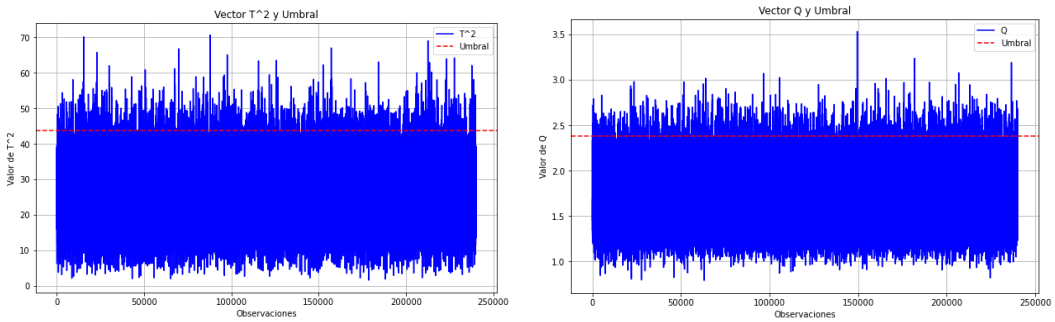


Figura 35. Estadísticos T^2 y Q en LSTM. Funcionamiento normal

Los umbrales de control se establecen en los percentiles 99 % para ambos estadísticos, delimitando así el rango de funcionamiento normal.

Por último, los parámetros adquiridos (umbrales, medias, matrices de covarianza y límites de normalización) y los modelos entrenados se guardan en archivos para asegurar que estén disponibles para la etapa siguiente, que es la detección de fallos.

4.4.2 Detección de Fallos

En primer lugar, se cargan los parámetros característicos obtenidos a lo largo del entrenamiento (como las medias y covarianzas del espacio latente, los límites superior e inferior de normalización y los umbrales de control para las estadísticas Q y T), así como también los modelos que fueron almacenados antes del autoencoder y su codificador correspondiente.

Se aplican los mismos límites utilizados en la fase de entrenamiento para normalizar los datos nuevos, lo que asegura la coherencia entre los dos conjuntos.

Las observaciones se reordenan después en secuencias temporales que tienen la misma longitud que la ventana empleada por la LSTM, para que así el modelo sea capaz de analizar cómo se comportan las variables de manera dinámica dentro de cada intervalo.

El autoencoder recrea cada una de las secuencias de entrada y, con base en estas reconstrucciones, se vuelven a calcular las estadísticas de control, de los fallos 6, 12 y 15 tal y como se comenta en el apartado anterior (Figura 36, Figura 37 y Figura 38).

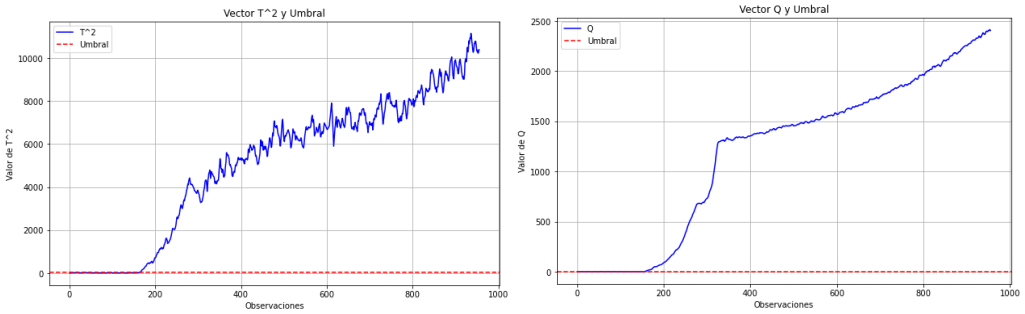


Figura 36. Detección del fallo IDV6 con LSTM. Estadísticos T^2 y Q

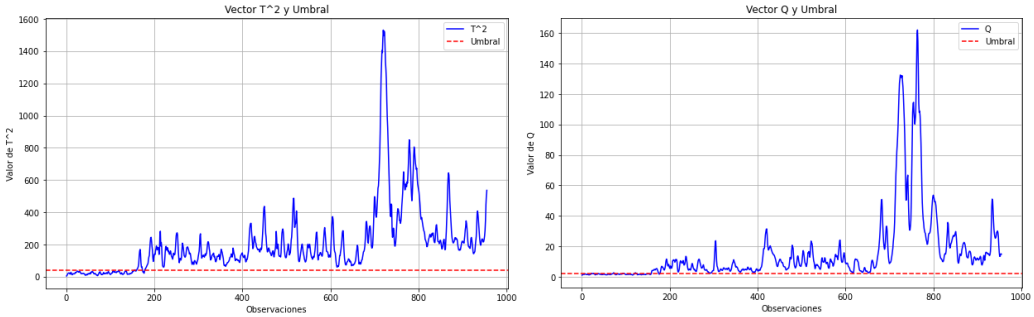


Figura 37. Detección del fallo ID12 con LSTM. Estadísticos T^2 y Q

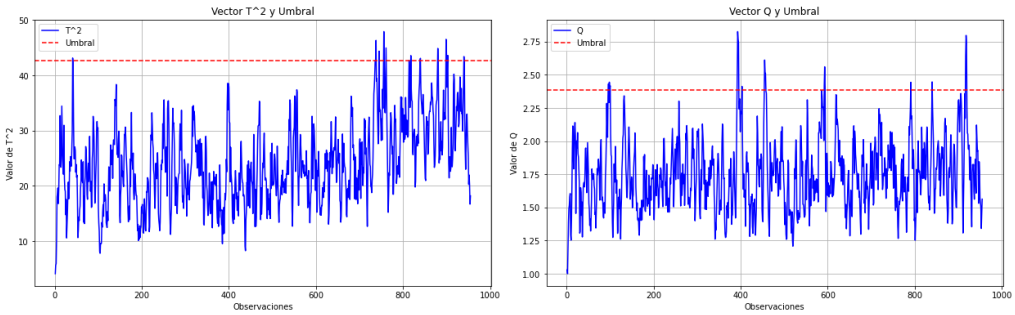


Figura 38. Detección del fallo IDV15 con LSTM. Estadísticos T^2 y Q

Para confirmar la presencia de un fallo se emplean los mismos criterios que en los apartados anteriores y se obtienen las Tabla 12, 13 y 14.

El RAE no solo detecta desviaciones instantáneas, sino también patrones anómalos en el tiempo. Una secuencia se considera anómala si su error de reconstrucción acumulado excede los límites derivados de los datos de entrenamiento, es por esto que Q suele ser más eficiente a la hora de detectar fallos en este tipo de autoencoder.

En última instancia, realizamos la identificación de fallos sobre el conjunto estudiado y en los dos que han sido detectados, representados en dos gráficos de barras en la Figura 39.

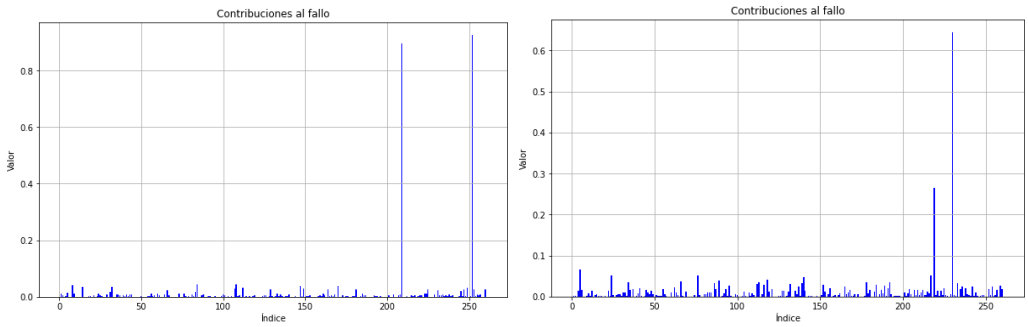


Figura 39. Contribuciones al fallo en LSTM IDV6 e IDV12

En este caso, esta representación evidencia las variables que han sido más contribuyentes en el fallo, en el caso del IDV6, las variables 36 y 42 han sido quienes más han influido, mientras que en el caso del IDV12, la variable que más ha contribuido ha sido la 40 seguida de la 37.

4.4.3 Resultados

El fallo 6 (Figura 36) se detecta de forma similar los métodos anteriores, los tiempos de observación son similares, sin embargo, las alarmas que detecta el autoencoder LSTM son similares al autoencoder.

En el fallo 12 (Figura 37) los dos estadísticos del autoencoder LSTM han detectado más rápido los fallos que los dos modelos anteriores y además con un porcentaje de alarmas superior

Por último, el fallo 15 (Figura 38) ahora vuelve a no ser detectado por el estadístico Q, al igual que en PCA vemos que el autoencoder es superior a estos dos en este caso.

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas(%)**	**t_Fallo(obs)**	**FalsasA(%)**	**Alarmas(%)**	**t_Fallo(obs)**
1	1,25	99,87	161	0	99,75	162
2	2,5	98,99	168	1,25	98,74	170
3	1,25	1,26	**No Detect.**	1,875	0,75	**No Detect.**
4	0	7,67	**No Detect.**	6,875	100,00	156
5	1,25	25,66	168	6,25	23,40	157
6	0	99,75	162	2,5	100,00	156
7	2,5	51,95	157	2,5	100,00	156
8	0	97,74	178	0	97,99	176
9	1,25	1,13	**No Detect.**	0	0,50	**No Detect.**
10	0	37,23	262	0,625	21,38	257
11	0,625	18,11	444	0	87,30	162
12	0,625	99,37	160	3,125	99,50	158
13	0	94,21	206	0	95,47	196
14	1,875	13,96	**No Detect.**	3,125	100,00	157
15	0,625	1,64	**No Detect.**	2,5	2,01	**No Detect.**
16	2,5	18,24	353	1,875	13,96	396
17	1,25	80,13	184	0	95,72	178
18	0	88,55	252	5	90,06	239
19	0	1,13	**No Detect.**	0	49,81	340
20	0	36,86	245	0	50,44	242
21	0	37,74	668	0,625	40,75	640

Tabla 12. detección de fallos mediante LSTM

Promedios LSTM			
	Falsas Alarmas (%)	**Alarmas (%)**	**Tiempo de detección (observaciones)**
T^2	0,83	48,15	251,20
Q	1,82	65,12	227,67

Tabla 13. Resumen Estadístico LSTM

Total Fallos LSTM	
T^2	15
Q	18

Tabla 14. Fallos detectados con LSTM

En vista a los resultados obtenidos, observamos que la red LSTM detecta menor número de fallos que el autoencoder pero mayor numero que PCA de forma global, vemos que los estadísticos siguen la misma dinámica que el método anterior, Q es superior en los dos tipos de autoecoders, pero T^2 es superior en PCA.

Por otra parte, si bien de forma general, el autoencoder ha detectado más fallos y más alarmas también ha detectado más falsas alarmas, donde destaca el LSTM por detectar un número mínimo de las mismas, especialmente en la estadística Q.

4.5 RAE distribuido

4.5.1 Entrenamiento

La manera de entrenar es igual para los tres escenarios de variables distribuidas (sesgo 1, sesgo 1.5 y sesgo 0.5), con la única diferencia en la conformación de los grupos que se logran a través del método Minimum Redundancy Maximum Relevance (mRMR), tal y como vimos en el apartado 2.8.1. Donde este sesgo, nos indica que umbral ponemos para decidir que variables entran en cada bloque. Por ejemplo, si el sesgo es igual a 1, hemos puesto como umbral el valor medio del índice mRMR (I_{mRMR}), es decir, si entre dos variables x_i y x_j el índice I_{mRMR} es mayor que el umbral, la variable x_j entra en el bloque definido por la variable x_i, si es menor no entra. Y esto se hace con todas las variables.

En todos los casos, las variables elegidas se subdividen en distintos subgrupos que posibilitan el desarrollo de un sistema de monitorización distribuida. En este sistema, cada submodelo recoge dinámicas particulares de una porción específica de la planta, Tablas 15, 16 y 17.

Se lleva a cabo una selección inicial de 240,000 muestras libres de fallos, suprimiendo las columnas que no aportan información, al igual que en los métodos anteriores, ya que necesitamos eliminar la mayor cantidad de ruido.

Los índices de las variables se agrupan en función de la estructura que corresponde a cada sesgo (1, 1.5 o 0.5) y se crean matrices normalizadas $X_i n$ para cada grupo i, según el resultado del mRMR, y se va a generar ahora un autoencoder recurrente por cada bloque.

La función de entrenamiento de cada autoencoder LSTM es idéntica a la empleada en el apartado 3, con la diferencia de que entrenamos un modelo de autoencoder con diferente número de variables para cada bloque, por lo tanto, hemos de emplear una configuración neuronal diferente para cada caso.

Para unificar el código y emplear unos números máximo y mínimo de neuronas se han establecido los criterios para la arquitectura de todos los modelos de autoencoder LSTM definidos en la Tabla 18.

Sesgo 1	Variables
X1	1, 44, 15, 49, 12, 48, 30, 37, 45
X2	3, 4, 7, 8, 10, 11, 13, 16, 18, 19, 22, 25, 31, 35, 43, 47, 50
X3	5, 17, 42, 46, 52
X4	2, 9, 21, 51
X5	20, 27, 28, 33, 34, 36
X6	6, 23, 24, 29, 38, 39, 41
X7	14, 26, 32, 40

Tabla 15. Agrupación de variables obtenida para Sesgo = 1 en mRMR

Sesgo 1,5	Variables
X1	1, 44, 14, 40, 15, 45,49
X2	7, 8, 10, 11, 13, 16, 18, 19, 22, 25, 31, 35, 43, 47, 50
X3	5, 17, 46, 52
X4	2, 9 ,21, 42, 51
X5	20, 27, 26, 33, 36
X6	4, 6, 23, 24, 38, 39, 41
X7	3, 29, 30, 34, 37
X8	12, 26, 42, 38

Tabla 16. Agrupación de variables obtenida para Sesgo = 1,5 en mRMR

Sesgo 0,5	Variables
X1	1, 2, 9, 14, 39, 44, 51
X2	8, 31, 4, 37, 22, 10, 25, 6, 43, 11, 18, 35, 50, 3, 47, 19, 16, 41, 29,7, 21, 13, 33, 20
X3	5, 46, 52, 42, 34, 17
X4	36, 27, 15, 28, 45, 26, 40, 23, 30, 38, 49
X5	32, 23, 12, 48

Tabla 17. Agrupación de variables obtenida para Sesgo = 0,5 en mRMR

Capa	Nombre de capa	Tipo de capa	Dimensión	Función de activación	Tipo
0	Input	Entrada	(5, input_dim)	—	Entrada
1	encoded1	LSTM	[5,(max (input_dim / 1.3), 4)]	SELU	Encoder
2	encoded2	LSTM	(max (input_dim / 3), 2)	SELU	Encoder (hidden)
3	repeat_vector	RepeatVector	[5, (max (input_dim / 3), 2)]	—	Decoder
4	decoded1	LSTM	[5, (max (input_dim / 3), 2)]	SELU	Decoder
5	decoded2	LSTM	[5,(max (input_dim / 1.3), 4)]	SELU	Decoder
6	output	Dense	52	Lineal	Salida

Tabla 18. Arquitectura encoders DLSTM

Tras el entrenamiento se generan tantos autoencoders LSTM como grupos de variables tengamos por sesgo y se obtienen las series de estadísticos T^2 y Q para cada grupo y sus umbrales, los cuales emplearemos mas adelante para la detección por grupos.

Finalmente se aplica un esquema de inferencia bayesiana, tal y como se describe en el apartado 2.8.2, para combinar la información procedente de todos los bloques en una única decisión global sobre el estado del proceso. De esta manera obtenemos los BICS de T^2 y Q, los cuales son equivalentes a los estadísticos de los apartados anteriores y sus umbrales (que en este caso corresponde al valor 1-α, donde α es el grado de precisión deseado, en este caso 0.5), los cuales emplearemos más adelante en la detección de fallos, del mismo modo que en los apartados anteriores.

Entonces se almacenan todos los datos obtenidos en un fichero para la posterior detección, el cual incluye: los modelos de encoders y decoders, estadísticos y umbrales por grupo y BICs.

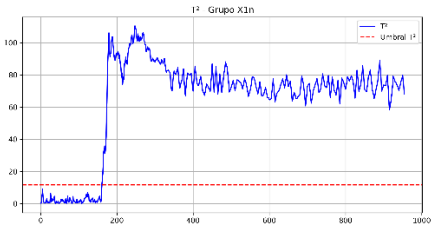
4.5.2 detección de Fallos

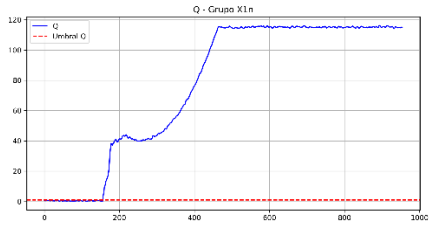
Se lleva a cabo la identificación de errores en los tres esquemas distribuidos mediante el empleo de los modelos que han sido entrenados previamente en cada conjunto de variables.

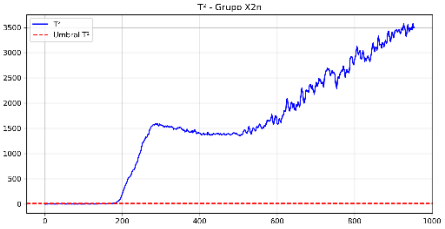
El propósito de este método es crear un mecanismo de diagnóstico que pueda combinar información proveniente de cada submodelo local y producir un índice total más resistente frente a interrupciones o errores de diferentes tipos.

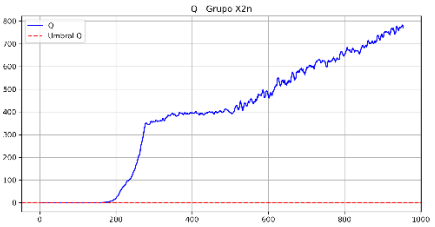
4.5.2.1 Detección de fallos por grupo

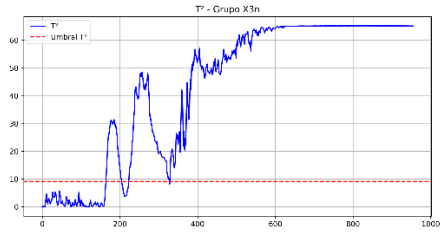
En primer lugar, realizamos la detección con los estadísticos de cada subgrupo para todos los fallos, este procedimiento es idéntico al realizado en los apartados anteriores, ya que la detección se realiza con los estadísticos T^2 y Q de cada subgrupo y sus umbrales en cada agrupación obtenida en mRMR por sesgo, representando la detección del fallo IDV6 obtenemos las figuras 40, 41 y 42.

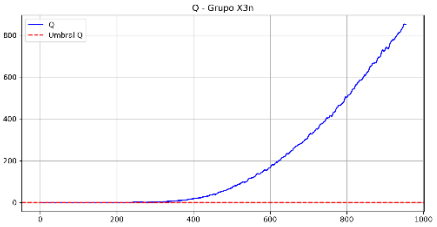

I² Grupo X1n
T²
Umbral I²

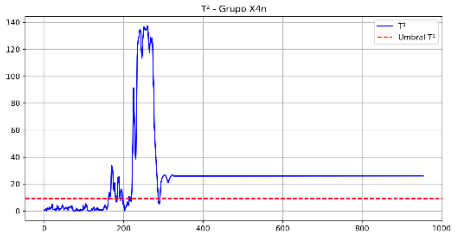

Q - Grupo X1n
Q
Umbral Q

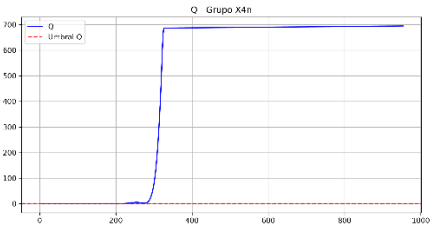

T² - Grupo X2n
T²
Umbral T²

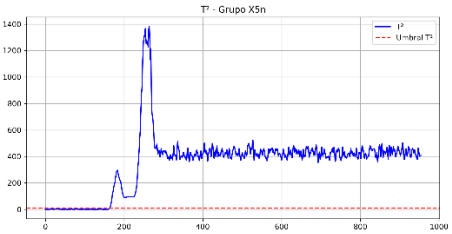

Q Grupo X2n
Q
Umbral Q

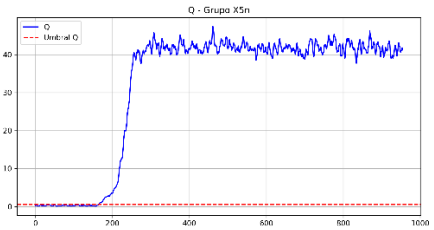

T² - Grupo X3n
T²
Umbral T²

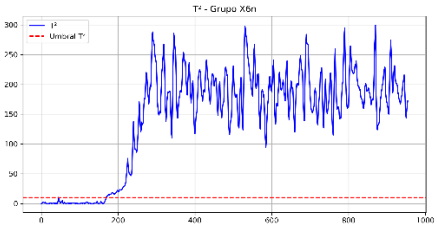

Q - Grupo X3n
Q
Umbral Q

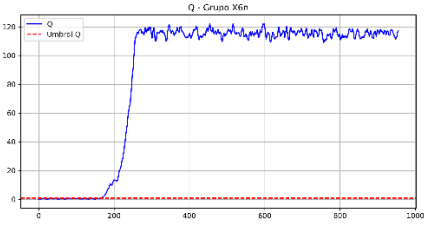

T² - Grupo X4n
T²
Umbral T²


Q Grupo X4n
Q
Umbral Q


T² - Grupo X5n
T²
Umbral T²


Q - Grupo X5n
Q
Umbral Q


T² - Grupo X6n
T²
Umbral T²


Q - Grupo X6n
Q
Umbral Q

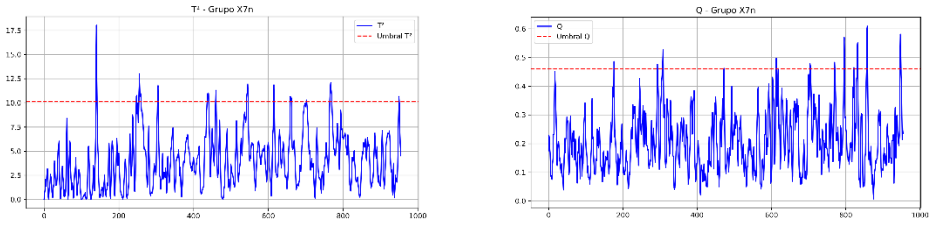
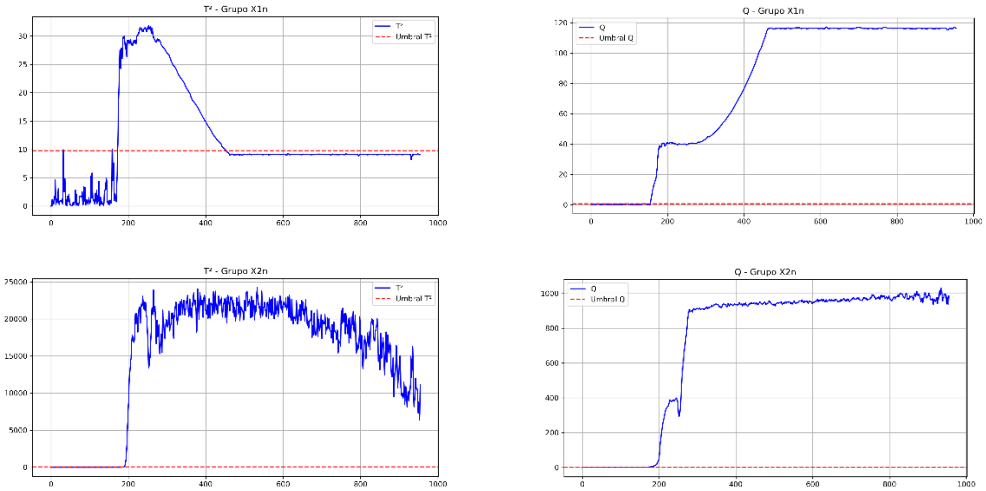


Figura 40. Estadísticos T^2 y Q fallo IDV6 (Sesgo=1)

Como se observa en este caso, la mayoría de los grupos detectan el fallo, pero el grupo 7 no, esto es debido a que no todas las variables contribuyen al fallo de la misma manera y al distribuir el método, conseguimos aislar que grupos presentan mayor susceptibilidad al fallo.



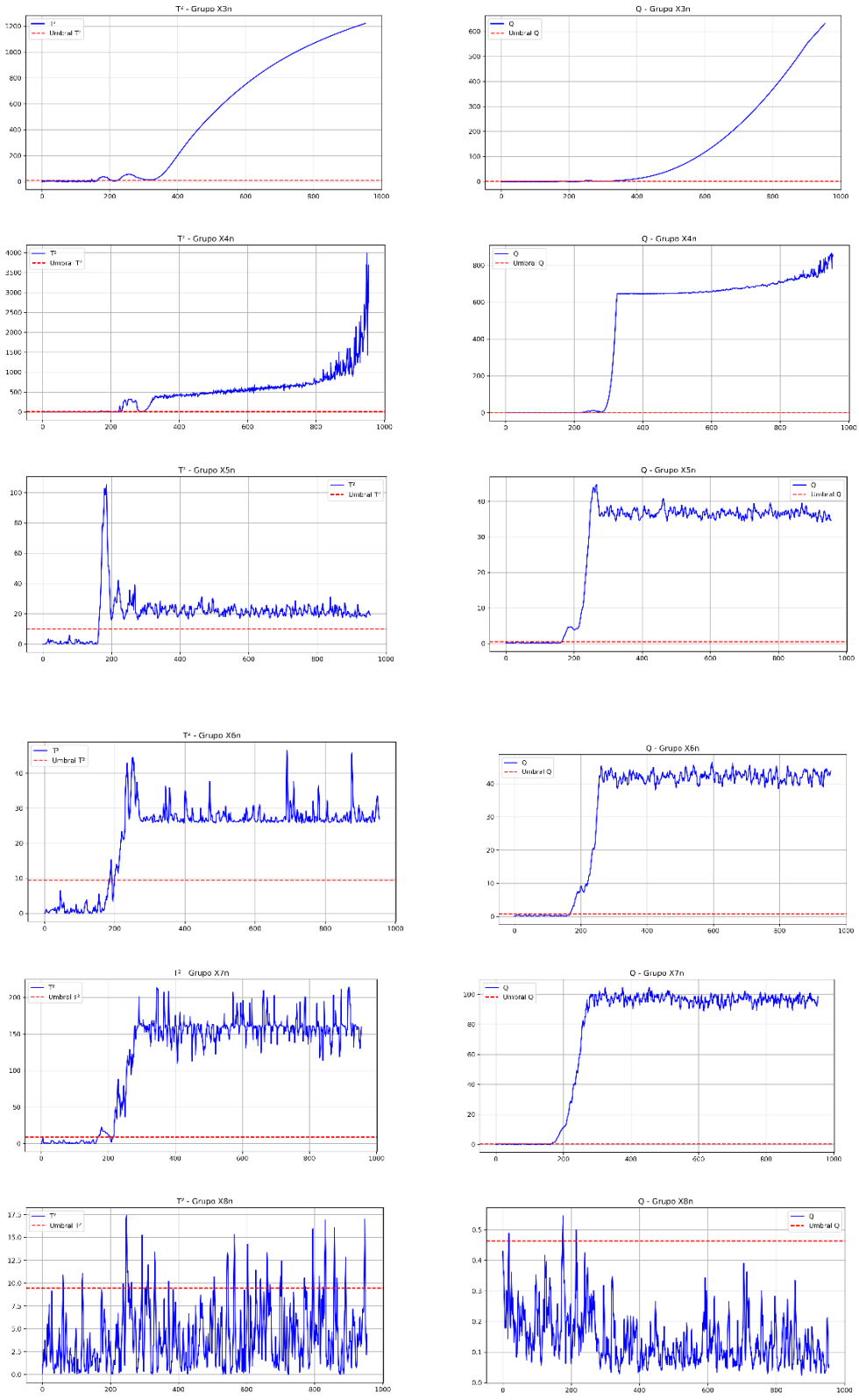


Figura 41. Estadísticos T^2 y Q fallo IDV6 (Sesgo=1,5)

En este caso vemos que todos los grupos detectan fallo menos el grupo 8, el cual, tal y como podemos observar en las Tabla 15 y 16 comparte variables con el grupo 7, el cual no detectaba fallo en la agrupación de sesgo 1.

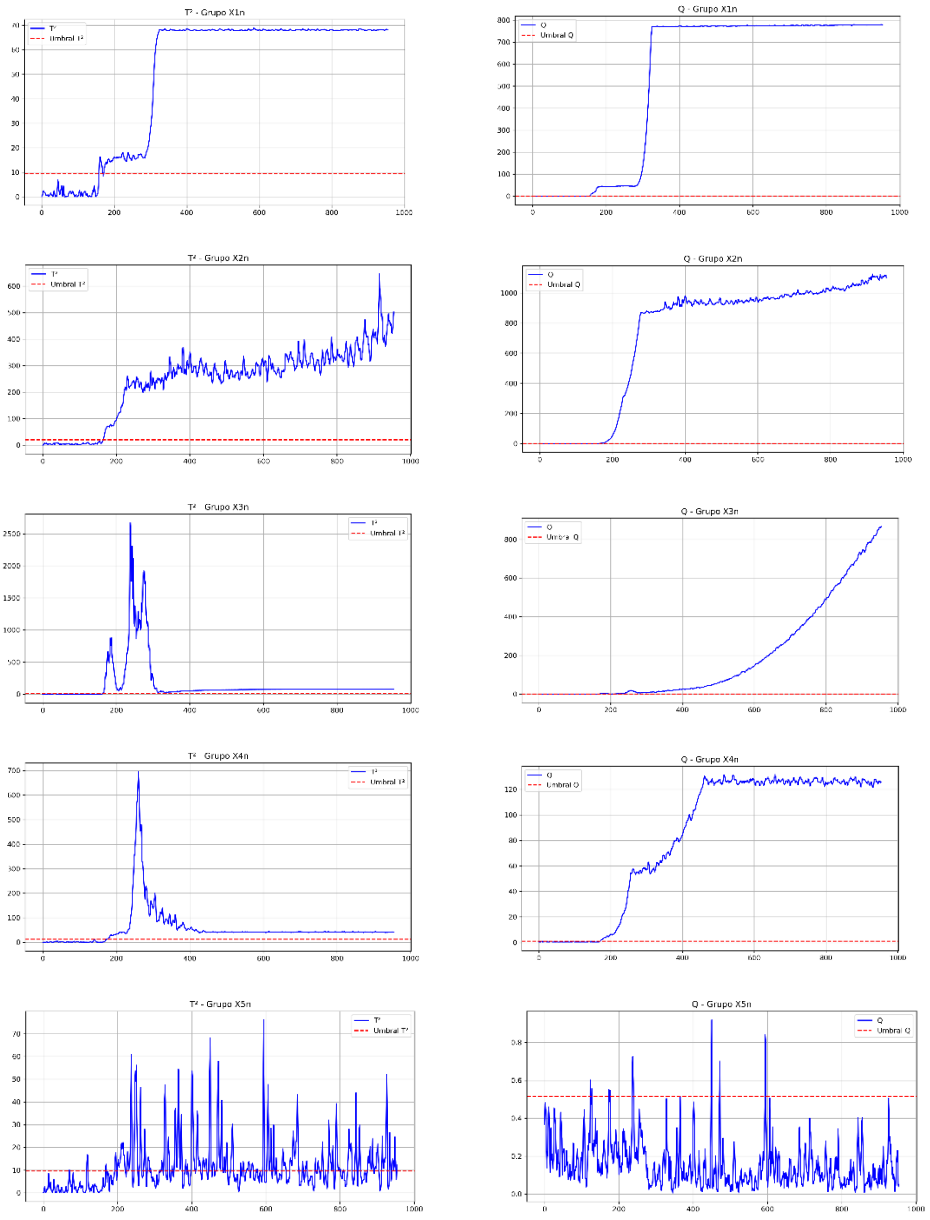


Figura 42. Estadísticos T^2 y Q fallo IDV6 (Sesgo=0,5)

En este caso es el grupo 5 el que no detecta fallo para el mismo ID de fallo, por lo tanto, vemos que los datos contenidos en el subgrupo son los únicos que no contribuyen al fallo.

Del mismo modo que en los apartados anteriores, se ha realizado el cálculo de las contribuciones al fallo para cada grupo de variables en cada sesgo y fallo, que se obtiene del cuadrado de los residuos.

Esto posibilita observar qué variables tienen mayor impacto en la anomalía detectada, lo cual ayuda a identificar la causa del error, como el grupo de variables es menor las gráficas representadas son mas esclarecedoras ya que permiten una mejor discriminación entre variables adyacentes.

Debido al gran número de gráficas obtenidas, se ha tomado como ejemplo el fallo IDV6, donde se ha detectado que el bloque 1 ha sido el mayor contribuyente a la producción del mismo, por ende, representando la contribución al fallo del bloque 1 para cada uno de los sesgos, obtenemos la Figura 43:

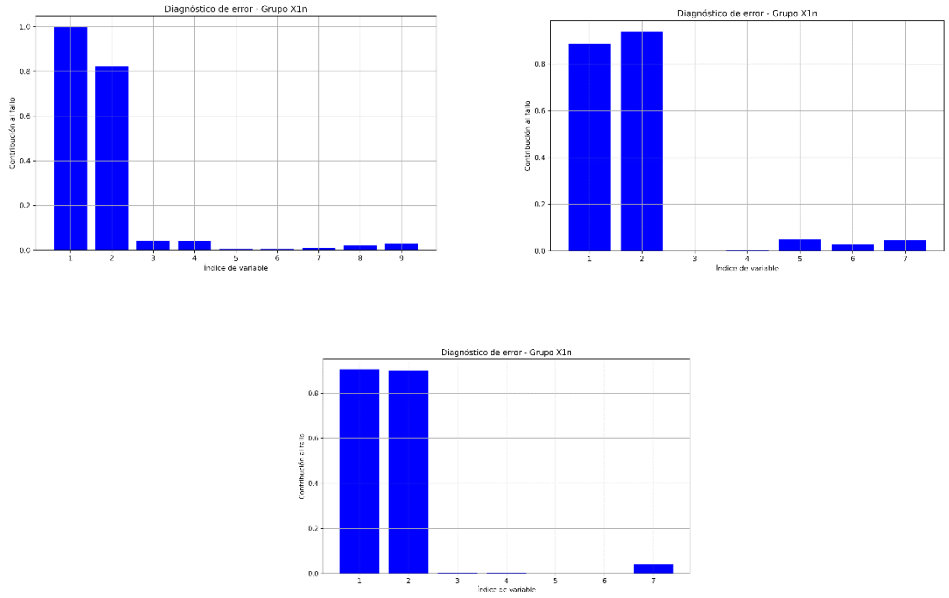


Figura 43. Contribuciones al fallo grupo X1 IDV6

Se observa claramente la importancia de las variables 1 y 2 en la producción del fallo, por lo tanto, podríamos concluir que son las causantes directas del mismo.

A continuación, presentamos la lista de los fallos detectados con los distintos sesgos (Tabla 19, 20 y 21), pudiendo observar que bloques contribuyen al fallo.

Grupo / Fallo	X1	X2	X3	X4	X5	X6	X7
1	SI	SI	SI	SI	SI	SI	SI
2	SI	SI	SI	NO	SI	SI	NO
3	NO	NO	NO	NO	NO	NO	NO
4	NO	NO	NO	SI	NO	NO	NO
5	SI	SI	SI	SI	SI	SI	NO
6	SI	SI	SI	SI	SI	SI	SI
7	SI	SI	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	SI	SI	SI
9	NO	NO	NO	NO	NO	SI	NO
10	SI	SI	SI	NO	SI	SI	NO
11	NO	NO	NO	SI	NO	NO	NO
12	SI	SI	SI	SI	SI	SI	SI
13	SI	SI	SI	SI	SI	SI	SI
14	NO	NO	NO	SI	NO	NO	NO
15	NO	SI	NO	NO	SI	NO	NO
16	NO	SI	SI	NO	SI	SI	NO
17	NO	SI	SI	SI	NO	NO	NO
18	SI	SI	SI	SI	SI	SI	SI
19	NO	NO	SI	SI	NO	NO	NO
20	NO	SI	SI	NO	SI	NO	NO
21	NO	SI	NO	SI	SI	NO	NO

Tabla 19. Detección por bloque sesgo 1

Observamos que hay fallos que sólo impactan a un grupo de variables y que solamente algunos bloques los detectan, como el 4, el 9 o el 11. En cambio, otros fallos, como el 6, el 7 y el 12, tienen un impacto más amplio en las variables del proceso.

Se nota, además, que algunos bloques identifican la mayor parte de los errores; en este caso, el bloque 2. En cambio, otros tienen una contribución más baja: por ejemplo, el bloque 7 solo detecta 7 fallos.

Es notable que el bloque 4, a pesar de contener solo 5 variables, sea de los más propensos a fallos (14, junto con el bloque 3, 4 y 5). Esto nos indica que las variables del cuarto bloque son muy relevantes para la detección.

Grupo / Fallo	X1	X2	X3	X4	X5	X6	X7	X8
1	SI	SI	SI	SI	SI	SI	SI	NO
2	SI	SI	SI	SI	SI	SI	SI	NO
3	NO	NO	NO	NO	NO	NO	NO	NO
4	NO	NO	NO	SI	NO	NO	NO	NO
5	SI	SI	SI	SI	SI	SI	SI	SI
6	SI	SI	SI	SI	SI	SI	SI	NO
7	SI	SI	SI	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	SI	SI	SI	SI
9	NO	NO	NO	NO	NO	NO	NO	NO
10	SI	SI	SI	NO	SI	SI	SI	NO
11	NO	NO	NO	SI	NO	NO	NO	NO
12	SI	SI	SI	SI	SI	SI	SI	SI
13	SI	SI	SI	SI	SI	SI	SI	SI
14	NO	NO	NO	SI	NO	NO	NO	NO
15	NO	NO	NO	NO	NO	NO	NO	NO
16	NO	SI	SI	NO	SI	SI	SI	NO
17	NO	SI	SI	SI	SI	SI	NO	NO
18	SI	SI	SI	SI	SI	SI	SI	SI
19	NO	NO	SI	SI	NO	NO	NO	NO
20	NO	SI	SI	NO	SI	NO	NO	NO
21	NO	SI	SI	NO	SI	NO	NO	NO

Tabla 20. Detección por bloque sesgo 1,5

En este caso la detección de fallos por grupos es similar al anterior, la principal diferencia reside en que el grupo 8 es quien detecta menos fallos en vez del 7 en el apartado anterior, sin embargo, ningún grupo ha sido capaz de detectar los fallos en los IDs 3, 9 y 15, a diferencia del anterior, donde el grupo 4 si ha detectado fallo en los ID 9 y 15.

Por último, con el sesgo 0,5, es decir un umbral por debajo del valor medio, vemos que el grupo 1 es quien detecta la mayor cantidad de fallos y el 5 la menor, también vemos que con respecto al sesgo 1 se deja de detectar el fallo 9.

Grupo / Fallo	X1n	X2n	X3n	X4n	X5n
1	SI	SI	SI	SI	SI
2	SI	SI	SI	SI	SI
3	NO	NO	NO	NO	NO
4	SI	NO	NO	NO	NO
5	SI	SI	SI	SI	SI
6	SI	SI	SI	SI	SI
7	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	SI
9	NO	NO	NO	NO	NO
10	SI	SI	SI	SI	NO
11	SI	NO	NO	NO	NO
12	SI	SI	SI	SI	SI
13	SI	SI	SI	SI	SI
14	SI	SI	NO	NO	NO
15	SI	NO	NO	NO	NO
16	SI	SI	SI	SI	NO
17	SI	SI	NO	NO	NO
18	SI	SI	SI	SI	SI
19	NO	NO	SI	NO	NO
20	SI	SI	SI	NO	NO
21	NO	SI	SI	NO	NO

Tabla 21. Detección por bloque sesgo 0,5

4.5.2.2 Detección de fallos global mediante BICs

Después, mediante la combinación de índices estadísticos individuales por medio de los Coeficientes de Influencia Bayesianos (BIC), realizaremos el mismo procedimiento que en los apartados anteriores

Los pasos que se describen a continuación son los que se siguen en el proceso integral de detección, aplicado a todos los escenarios de fallo. Primero, para cada uno de los 21 errores del Tennessee Eastman Process, se carga el archivo de datos correspondiente. Se lleva a cabo la normalización con los valores máximo y mínimos conseguidos en el periodo de entrenamiento, asegurando así la coherencia en relación al espacio en que fueron entrenados los autoencoders.

Luego, se extraen de cada conjunto solo las variables que forman parte del subconjunto definido por mRMR y se crean las secuencias temporales requeridas para la evaluación de los modelos LSTM.

Se examina cada subgrupo de manera independiente, empleando el autoencoder LSTM preparado para ese conjunto, el codificador relacionado y los parámetros estadísticos que se obtuvieron durante el entrenamiento.

A partir de las reconstrucciones, como ya se explicó en el apartado 4.4.2 se calculan los estadísticos para cada grupo, debido a que los autoencoders tienen la capacidad de generar secuencias de longitudes variadas (a causa del windowing), las series T^2 y Q de cada grupo se reducen a la longitud mínima compartida y se almacenan en una matriz.

Se utilizan los BIC sobre las matrices previamente mencionadas para determinar un indicador global de diagnóstico. Esta combinación, mediante la ponderación de la aportación de cada grupo según su comportamiento estadístico bajo condiciones normales, produce dos señales globales para la monitorización:

- BIC_T^2: combinación de T^2 distribuidos a través de un enfoque bayesiano.
- BIC_Q: combinación de Q distribuidos por medio de Bayes.

Los umbrales globales $uBIC\_T^2$ y $uBIC\_Q$ adquiridos en la fase de entrenamiento, se emplean como criterio final de decisión. Graficando el fallo IDV6 obtenemos las siguientes representaciones correspondientes a los casos de sesgo 1 (Figura 44), sesgo 1,5 (Figura 45) y sesgo 0,5 (Figura 46).

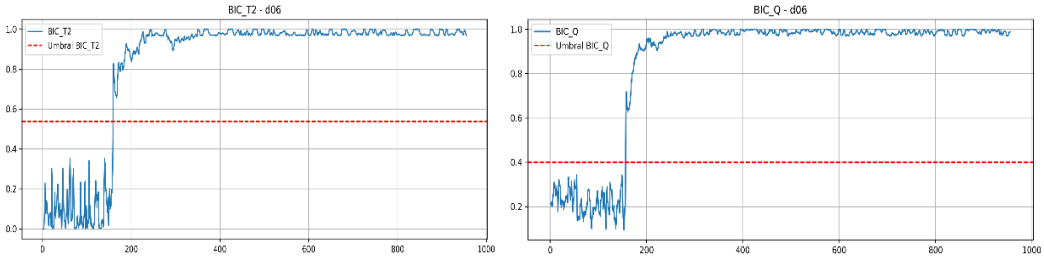


Figura 44. Detección del fallo IDV6 con DLSTM1. BIC_T2 y BIC_Q

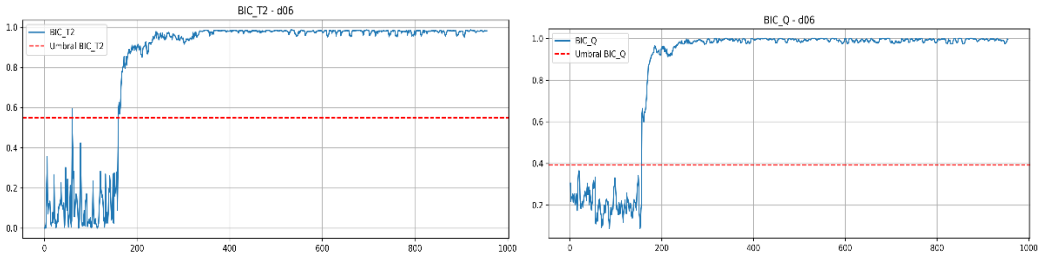


Figura 45. Detección del fallo IDV6 con DLSTM2. BIC_T2 y BIC_Q

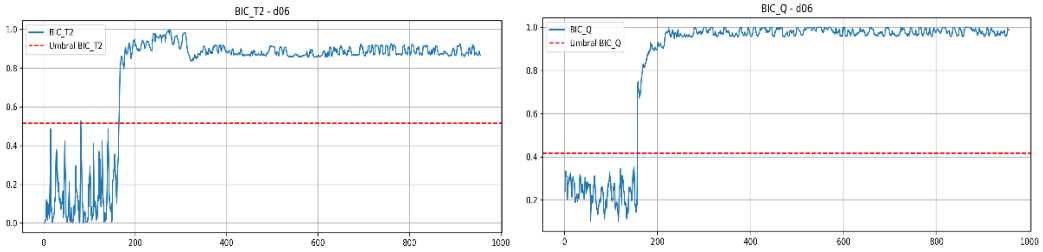


Figura 46. Detección del fallo IDV6 con DLSTM3. BIC_T2 y BIC_Q

En el caso del fallo IDV12, también representado en los métodos anteriores obtenemos la Figura 47, 48 y 49.

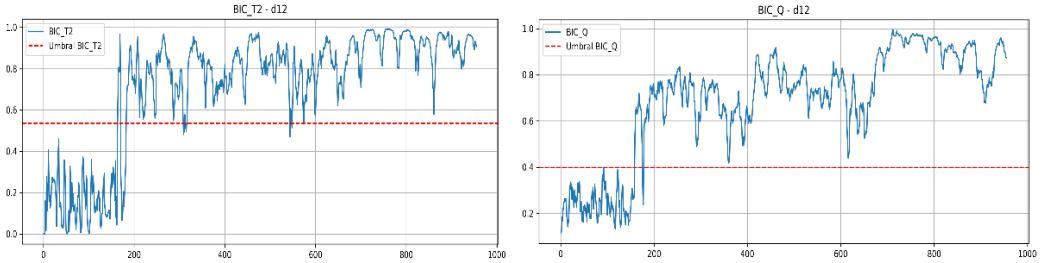


Figura 47. Detección del fallo IDV12 con DLSTM1. BIC_T2 y BIC_Q

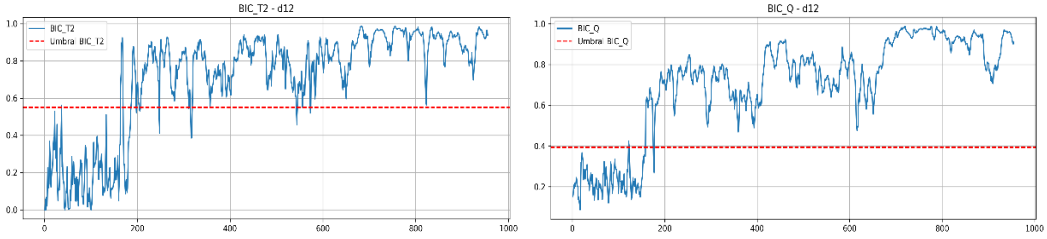


Figura 48. Detección del fallo IDV12 con DLSTM2. BIC_T2 y BIC_Q

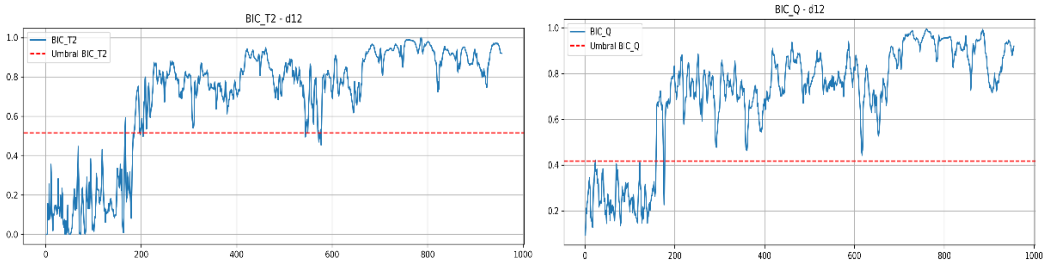


Figura 49. Detección del fallo IDV12 con DLSTM3. BIC_T2 y BIC_Q

Por último, representamos el fallo IDV15, también representado en todos los métodos empleados en el que no se detecta fallo, figuras 50, 51 y 52.

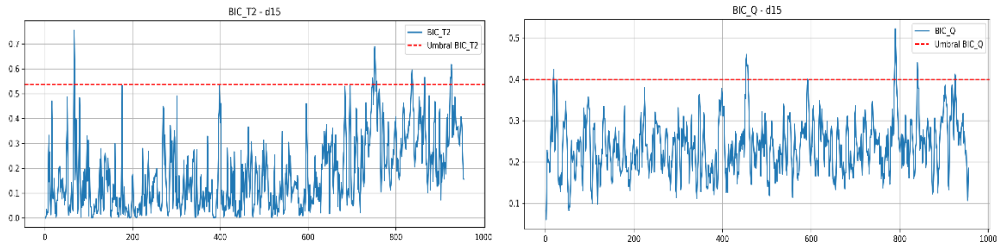


Figura 50. Detección del fallo IDV15 con DLSTM1. BIC_T2 y BIC_Q

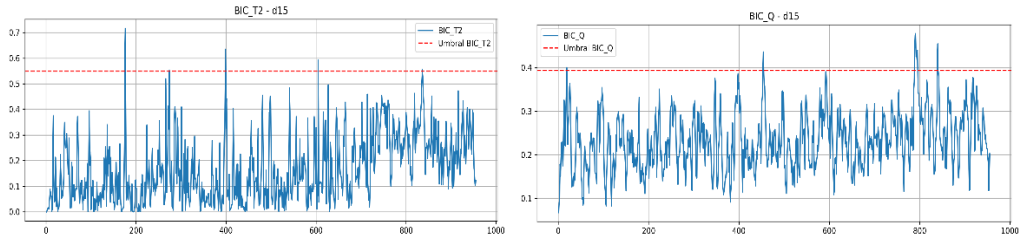


Figura 51. Detección del fallo IDV15 con DLSTM2. BIC_T2 y BIC_Q

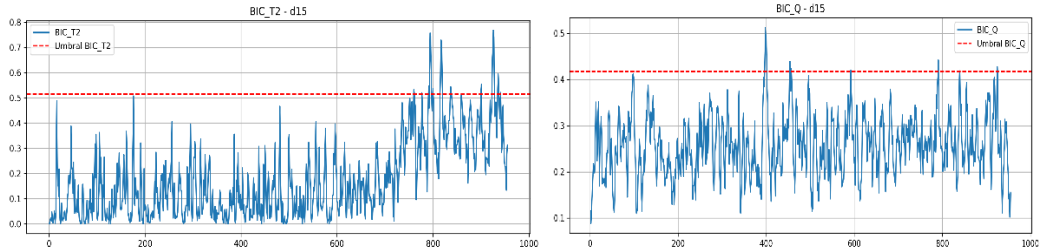


Figura 52. Detección del fallo IDV15 con DLSTM3. BIC_T2 y BIC_Q

Se han utilizado las tres mismas métricas estándar para cada error:

1. Falsas alarmas: Se examina el porcentaje de muestras en la ventana inicial normal (160 muestras) que sobrepasan el umbral sin error.
2. Alarmas identificadas: Se estima el porcentaje de muestras posteriores a la ventana normal que superan su umbral determinado.
3. Instante de detección: Se considera como el índice de la secuencia inicial de diez muestras sucesivas que sobrepasan el límite del BIC. Esto posibilita una detección más sólida y menos susceptible al ruido.

Estas métricas se han calculado tanto para BIC_T2 como para BIC_Q, obteniendo así seis indicadores por cada uno de los 21 fallos analizados como se ve en las Tablas 22 a 30.

4.5.3 Resultados

El fallo 6 (Figura 44, 45 y 46) se detecta de forma similar para todas las agrupaciones de variables y al modelo no distribuido

En el fallo 12 (Figura 47, 48 y 49) es detectado también de forma similar en todos los grupos y también de forma similar al modelo no distribuido

Por último, el fallo 15 (Figura 50, 51 y 52) ahora vuelve a no ser detectado por el estadístico Q, al igual que en PCA vemos que el autoencoder es superior a estos dos en este caso.

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas(%)**	**t_Fallo(obs)**	**FalsasA(%)**	**Alarmas(%)**	**t_Fallo(obs)**
1	1,875	99,245	166	1,25	99,874	161
2	3,125	98,994	168	0	98,742	170
3	1,875	0,377	**No Detect.**	1,25	1,006	**No Detect.**
4	1,875	32,83	496	2,5	100	156
5	0,625	22,39	171	1,875	30,189	157
6	1,25	100	158	2,5	100	156
7	1,875	38,113	157	2,5	100	156
8	0	97,736	178	0	97,862	177
9	1,875	0,881	**No Detect.**	0,625	1,132	**No Detect.**
10	0,625	29,308	300	0	24,025	258
11	2,5	54,591	163	0,625	91,447	162
12	0	96,981	182	1,25	99,748	158
13	0,625	94,843	204	0	95,094	199
14	0,625	41,887	**No Detect.**	2,5	100	156
15	1,25	2,013	**No Detect.**	1,25	1,761	**No Detect.**
16	0,625	13,836	465	0	20,881	362
17	0,625	79,874	179	0,625	96,981	177
18	0,625	89,811	248	0	90,189	238
19	0	1,635	**No Detect.**	0,625	58,239	173
20	0	33,711	244	0	39,623	245
21	0	32,075	708	0	41,258	641

Tabla 22. Detección de Fallos mediante DLSTM1

Total Fallos DLSTM1	
T^2	16
Q	18

Tabla 23. Fallos detectados por DLSTM1

	Promedios DLSTM1		
Estadístico	**Falsas Alarmas (%)**	**Alarmas (%)**	**Tiempo de detección (observaciones)**
T^2	1,04	50,53	261,69
Q	0,92	66,10	216,78

Tabla 24. Resumen estadístico DLSTM1

Se observa que la detección de fallos en el modelo distribuido, empleando los BICs es similar al autoencoder LSTM, los tiempos de detección, numero de fallos detectados y porcentaje de alarmas son muy similares.

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas(%)**	**t_Fallo(obs)**	**FalsasA(%)**	**Alarmas(%)**	**t_Fallo(obs)**
1	0,625	99,497	164	0	99,874	161
2	1,875	98,868	169	2,5	99,119	167
3	1,875	1	**No Detect.**	0	1,258	**No Detect.**
4	0,625	18,868	497	2,5	100	156
5	1,25	24,025	175	1,875	33,836	157
6	1,25	100	159	2,5	100	156
7	0,625	38,113	160	2,5	100	156
8	0	97,358	179	0	97,736	178
9	0,625	0,252	**No Detect.**	0,625	0,755	**No Detect.**
10	0	25,66	258	0	33,585	206
11	1,875	30,818	166	0,625	90,566	161
12	0,625	95,346	186	2,5	99,748	158
13	0	94,969	204	0	94,969	199
14	1,875	14,088	**No Detect.**	3,125	100	156
15	0	0,629	**No Detect.**	0,625	1,509	**No Detect.**
16	0,625	12,956	468	0,625	25,535	365
17	1,25	76,478	179	1,25	96,604	177
18	1,25	88,931	251	0	90,314	237
19	3,125	5,031	**No Detect.**	0	60,377	173
20	1,875	50,566	237	0	40,126	244
21	0	38,742	673	0	40,503	642

Tabla 25. Detección de Fallos mediante DLSTM2

Total Fallos DLSTM2	
T^2	16
Q	18

Tabla 26. Fallos detectados por DLSTM2

	Promedios DLSTM2		
Estadístico	**Falsas Alarmas (%)**	**Alarmas (%)**	**Tiempo de detección (observaciones)**
T^2	1,01	48,20	257,81
Q	1,01	66,97	213,83

Tabla 27. Resumen estadístico DLSTM2

En el caso del grupo de sesgo 0,5 (DLSTM3) se observa un menor número de detecciones que en los dos anteriores, esto puede deberse a que las agrupaciones de variables tienen una menor correlación entre ellas dentro del proceso, por lo tanto, al separarlas del resto de variables limitan el aprendizaje de la red neuronal.

	Detectado por T^2			Detectado por Q		
Fallo	**FalsasA (%)**	**Alarmas(%)**	**t_Fallo(obs)**	**FalsasA(%)**	**Alarmas(%)**	**t_Fallo(obs)**
1	0,625	99,748	162	5	100	160
2	3,75	98,868	169	0	98,868	170
3	0	1,006	**No Detect.**	0	0,126	No Detect.
4	1,875	0,377	**No Detect.**	2,5	100	156
5	1,875	25,031	168	1,875	27,296	157
6	0,625	99,497	164	2,5	100	156
7	2,5	42,013	157	2,5	100	156
8	0	97,61	179	0	97,987	176
9	0	0,377	**No Detect.**	1,25	1,635	**No Detect.**
10	0,625	33,459	261	0	18,994	262
11	0	1,761	**No Detect.**	1,25	90,314	162
12	0	95,975	184	1,875	99,623	158
13	0	94,34	205	0	95,346	201
14	1,875	1,509	**No Detect.**	2,5	100	157
15	0	3,648	**No Detect.**	0	1,635	**No Detect.**
16	3,125	13,962	466	0,625	15,723	365
17	1,25	57,233	187	0	96,101	179
18	0	89,182	252	1,25	90,063	239
19	0	1,635	**No Detect.**	0	54,717	172
20	1,25	49,937	236	0	39,623	246
21	2,5	35,472	676	0,625	41,006	642

Tabla 28. Detección de Fallos mediante DLSTM3

Total Fallos DLSTM3	
T^2	14
Q	18

Tabla 29. Fallos detectados DLSTM3

	Promedios DLSTM3		
Estadístico	**Falsas Alarmas (%)**	**Alarmas (%)**	**Tiempo de detección (observaciones)**
T^2	1,04	44,89	247,57
Q	1,13	65,19	217,44

Tabla 30. Resumen Estadístico DLSTM3

Capítulo 5. Conclusiones y trabajo futuro

5.1 Conclusiones

El presente trabajo ha abordado la monitorización de procesos industriales mediante técnicas de detección de fallos basadas en datos, utilizando el proceso Tennessee-Eastman (TE) como caso de referencia. A lo largo del estudio se han comparado métodos clásicos y modernos con el objetivo de evaluar su eficacia en la identificación de anomalías.

Método	Falsas Alarmas T^2 (%)	Alarmas (%) T^2
PCA	1,28	57,72
Autoencoder	0,36	49,35
LSTM	0,83	48,15
DLSTM (Sesgo=1)	1,04	50,53
DLSTM (Sesgo=1,5)	1,01	48,20
DLSTM (Sesgo=0,5)	1,04	44,89

Tabla 31. Sumario de alarmas detectadas por T

Método	Falsas Alarmas Q (%)	Alarmas (%) Q
PCA	3,54	67,14
Autoencoder	3,30	73,42
LSTM	1,82	65,12
DLSTM (Sesgo=1)	0,92	66,10
DLSTM (Sesgo=1,5)	1,01	66,97
DLSTM (Sesgo=0,5)	1,13	65,19

Tabla 32. Sumario de alarmas detectadas por Q

Método	Fallos detectados T^2	Fallos detectados Q
PCA	17	17
Autoencoder	14	20
LSTM	15	18
DLSTM (Sesgo=1)	16	18
DLSTM (Sesgo=1,5)	16	18
DLSTM (Sesgo=0,5)	14	18

Tabla 33. Sumario de fallos detectadas por método

Los resultados obtenidos, reflejados en las Tablas 31, 32 y 33 pueden resumirse en los siguientes puntos:

1. PCA demostró ser un método eficaz en la reducción de dimensionalidad y en la detección de fallos lineales, pero su capacidad se ve limitada en presencia de relaciones no lineales o de carácter dinámico. Aun así, su bajo coste computacional lo hace un método muy a tener en cuenta a la hora de detectar fallos, es por eso por lo que hoy en día se sigue empleando en numerosos procesos industriales.

2. El autoencoder clásico superó al PCA en escenarios con dependencias complejas, logrando una mejor reconstrucción de los datos y una mayor sensibilidad frente a ciertos fallos. Sin embargo, su ausencia de memoria temporal restringe su aplicación en procesos con secuencias dependientes.

3. Los RAE (Recurrent Autoencoders), introducidos a continuación, ofrecieron un rendimiento ligeramente inferior al autoencoder de capas densas en la detección de fallos con evolución temporal, si bien este es capaz de capturar patrones dinámicos que los enfoques anteriores no conseguían modelar con precisión, no ha conseguido los resultados esperados aunque podemos destacar que ha sido más preciso a la hora de detectarlos, dado que las falsas alarmas producidas por este método han sido menores.

4. La extensión mediante RAE distribuidos ha permitido escalar el modelo a sistemas de gran dimensionalidad, aportando interpretabilidad y facilitando la localización de fallos en bloques específicos de variables. Este enfoque distribuye la complejidad computacional y mejora la robustez del diagnóstico. Si bien los resultados esperados no han sido los obtenidos, ha obtenido resultados globales similares al LSTM no distribuido, con la ventaja de acotar los grupos de variables más influyentes en el fallo.

En conjunto, se concluye que la incorporación de modelos recurrentes y de esquemas distribuidos constituye una mejora sustancial respecto a los enfoques tradicionales, contribuyendo a aumentar la fiabilidad, la seguridad y la eficiencia en la operación de procesos industriales.

5.2 Trabajo futuro

A partir de los resultados obtenidos, se plantean diversas líneas de investigación orientadas a ampliar y consolidar las aportaciones realizadas:

1. Mejora de arquitecturas: RAE: Investigar configuraciones más complejas o mixtas (fusionando CNN y RNN) o mediante RVAE para optimizar la habilidad de representación de secuencias.

2. Puesta en práctica en tiempo real: trasladar los modelos a ambientes industriales con limitaciones de recursos computacionales y latencia, analizando su factibilidad práctica.

3. Integración de modelos: Con el fin de lograr diagnósticos más exhaustivos, fusionar los RAE con otras metodologías de detección, como los métodos basados en grafos o los modelos probabilísticos.

4. Generalización a otros procesos: verificar la solidez y la capacidad de transferir los resultados mediante la validación del método en otras plantas piloto y procesos industriales que no sean el Tennessee-Eastman.

5. Monitoreo explicable: analizar métodos de interpretabilidad que posibiliten entender cómo los modelos identifican anomalías, lo que favorece la adopción en el sector industrial y mejora la confianza de los trabajadores.

Bibliografía

[1] American Society for Quality (ASQ), "History of Quality." [Online]. Available: https://asq.org/quality-resources/history-of-quality. [Último acceso: 3-Sep-2025].

[2] M. P. Crosby, *The History of Quality Management*. Juran Institute, 2019. [Online]. Available: https://www.juran.com/blog/quality-management-system/ [Último acceso: 3-Sep-2025].

[3] W. A. Shewhart, *Economic Control of Quality of Manufactured Product*. New York, NY, USA: D. Van Nostrand Company, 1931. [Online]. Available: https://archive.org/details/economiccontrolo00shew. [Último acceso: 3-Sep-2025].

[4] P. D. Houston, "The Quality Revolution in Japan: The Contributions of Deming and Juran," *Peoria Magazines*, 2018. [Online]. Available: https://www.peoriamagazines.com/article/quality-revolution-japan-contributions-deming-and-juran. [Último acceso: 6-Sep-2025].

[5] https://blog.softexpert.com/es/que-es-ciclo-pdca/

[6] Juran Institute, "History of Joseph M. Juran." [Online]. Available: https://www.juran.com/about-us/dr-jurans-history/. Último acceso: 6-Sep-2025].

[7] International Organization for Standardization (ISO), "ISO 9000 Quality management." [Online]. Available: https://www.iso.org/iso-9001-quality-management.html. [Último acceso: 7-Sep-2025].

[8]https://corporatefinanceinstitute.com/resources/management/total-quality management-tqm/

[9] R. Karapetrovic, "ISO 9000: Evolution and Future Directions," *Quality Management Journal*, vol. 22, no. 1, pp. 30–40, 2015.

[10] H. F. Dodge and H. G. Romig, *Sampling Inspection Tables: Single and Double Sampling*. New York, NY, USA: Wiley, 1959.

[11] ScienceDirect, "Quality Control – An Overview." [Online]. Available: https://www.sciencedirect.com/topics/engineering/quality-control. [Último acceso: 18-Sep-2025].

[12] M. S. Phadke, *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[13] Carro Paz, R. & González Gómez, D. "Control Estadístico de Procesos", Universidad Nacional de Mar del Plata, recuperado en: https://nulan.mdp.edu.ar/id/eprint/1617/1/12_control_estadistico.pdf

[14] J. F. MacGregor and T. Kourti, "Statistical process control of multivariate processes," *Control Engineering Practice*, vol. 3, no. 3, pp. 403–414, 1995.

[15] S. W. Mason and N. D. Young, *Multivariate Statistical Process Control with Industrial Applications*. Philadelphia, PA, USA: ASA-SIAM, 2002.

[16] H. Hotelling, "Multivariate quality control—illustrated by the air testing of sample bombsights," in *Techniques of Statistical Analysis*, C. Eisenhart, M. W. Hastay, and W. A. Wallis, Eds. New York, NY, USA: McGraw-Hill, 1947, pp. 111–184.

[17] E. Jackson, *A User's Guide to Principal Components*. New York, NY, USA: Wiley, 1991.

[18] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002.

[19] Charles Gauvin. Distances and outlier detection. 2021. url: https://www.charlesgauvin.ca/post/distances-and-outlier-detection/ [Ultimo acceso: 10-10-2025]

[20] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 11, pp. 559–572, 1901.

[21] Davide Massidda. Multivariate Process Control by Principal Component Analysis Using T^2 and Q errors. 2023. url: https://towardsdatascience.com/multivariate-process-control-by-principal-component-analysis-using-t%C2%B2-and-q-errors-c94908d14b04 (Ultimo Acceso: 13-10-2025).

[22] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[23] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[25] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15 of *JMLR: W&CP 15*, Fort Lauderdale, FL, USA, 2011, pp. 315–323.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[28] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson, 2009

[29] A. Zhang, Z. C. Lipton, M. Li y A. J. Smola, «Dive into Deep Learning,» [En línea]. Available: https://d2l.ai/chapter_recurrent-modern/lstm.html. [Último acceso: 14 10 2025].

[30] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[32] K. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

[33] J. A. Barreto, E. L. Lima, y F. C. F. Silva, “A Bayesian inference approach for multivariate process monitoring using decentralized statistics,” *Journal of Process Control*, vol. 24, no. 8, pp. 1253–1263, 2014.

[34] Z. Ge, Z. Song, y F. Gao, “Review on data-driven modeling and monitoring for plant-wide industrial processes,” *Chemometrics and Intelligent Laboratory Systems*, vol. 171, pp. 16–25, 2017

[35] J. J. Downs and E. F. Vogel, “A plant-wide industrial process control problem,” *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993, doi: 10.1016/0098-1354(93)80018-I.

[36] 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,” *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.

[37] C. Aldrich, *Process Fault Diagnosis for Continuous Dynamic Systems Over Multivariate Time Series*, in C.-K. Ngan (ed.), Rijeka, Croatia: IntechOpen, 2019, ch. 1, doi: 10.5772/intechopen.85456.

[38] H. Chen, P. Tiňo, and X. Yao, “Cognitive fault diagnosis in Tennessee Eastman Process using learning in the model space,” *Computers & Chemical Engineering*, vol. 67, pp. 33–42, 2014, doi: 10.1016/j.compchemeng.2014.03.015.

[39] Massachusetts Institute of Technology – Braatz Group, “Braatz Group Data.” [Online]. Available: http://web.mit.edu/braatzgroup/links.html. [Último acceso: 30-10-2025].

[40] Harvard University, “Harvard Dataverse Dataset,” doi: 10.7910/DVN/6C3JR1. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6C3JR1. [Último acceso: 30-10-2025].

ChatGPT (OpenAI 2023) se utilizó para generar ideas iniciales y refinar la redacción de este trabajo. ChatGPT. (GPT-4). OpenAI. 27 de mayo de 2025. [En línea]. 2023. Disponible en: https://chatgpt.com