Departamento de Ingeniería Informática
y Ciencias de la Computación
Universidad de Concepción

«Monitoreo de procesos de conversión de energía mediante métodos de aprendizaje estadístico»

por
«Marcel Ignacio Gutiérrez Gutiérrez»

Memoria presentada para la obtención del título de
Ingeniero Civil Informático

Profesor Guía: «Hugo Garcés Hernández»

Concepción, «Enero, 2025»

Esta memoria fue patrocinada por ANID
a través del proyecto Fondecyt regular 1220903

Dedicatoria

Me gustaría dedicar esta memoria de título a mi madre, abuela y especialmente a mi padre (quién ya no se encuentra con nosotros), por el amor, paciencia y apoyo que siempre me han dado a lo largo de mi vida.

Agradecimientos

Quiero expresar mis más sinceros agradecimientos a las siguientes personas y entidades:

A mi tío Cristian, por el apoyo y palabras, que me ha dado no solo durante la producción de esta memoria, pero a lo largo de mi vida también.

Al profesor Enrique, por ser quién me ayudo a conocer a mi profesor guía.

Al profesor Hugo, por entregarme un tema que trabajar, más el apoyo y paciencia que me ha dado como profesor guía, para ayudarme a terminar esta memoria bajo las circunstancias personales en las que me encontraba.

Al proyecto Fondecyt regular 1220903 bajo el nombre de "Combining machine learning and data analytics for advances in process control" por patrocinar esta memoria.

Sumario

En el presente documento se expone el trabajo realizado por el alumno Marcel Ignacio Gutiérrez Gutiérrez para su Memoria de Titulo de nombre "Monitoreo de procesos de conversión de energía mediante métodos de aprendizaje estadístico", en donde el alumno presenta una serie de modelos realizados en lenguaje Python (PCA, PLS, KPCA y KPLS) aplicados sobre un contexto real (Paneles Fotovoltaicos) con propósitos comparativos. Durante el documento, el alumno explica las características del dataset utilizado como el procesamiento correspondiente para su uso, el funcionamiento general de los modelos y las métricas utilizadas, en conjunto a la estructura general de los códigos implementados, finalmente seguido de pruebas en donde se aplica anomalías artificiales sobre el dataset (Drift, Noise, Packet Loss, Retained Values y Packet Drop) para probar la eficacia de los modelos con respecto a la detección de fallas.

In the following document it's presented the work carried out by the student Marcel Ignacio Gutiérrez Gutiérrez for his college degree which it's named "Monitoring of energy conversion processes through the use of statistical learning methods". In this document, the student presents a series of models written in Python language (PCA, PLS, KPCA and KPLS) which has been applied on a real case (Photovoltaic Panels, most commonly known as solar panels) in order to compare their perfomances. During the document, the student explains the dataset and it's features as well as the preprocessing made on the dataset, it also explains the general functioning of the implemented methods together with the metrics utilized for this work, in addition to the structure of the codes produced. Lastly, the student presents the results obtained by the models and the tests performed using artificial anomalies (Drift, Noise, Packet Loss, Retained Values y Packet Drop) in order to check the efficacy and perfomance of the models regarding their ability for fault detection.

Tabla de Contenidos

Índice de tablas

Índice de figuras

1. Introducción

Dentro del área industrial, un aspecto de crucial importancia radica en el monitoreo de procesos, lo que permite que las industrias verifiquen la correcta ejecución de sus procesos, como también que las condiciones de trabajo sean seguras para las personas e infraestructuras involucradas. También ayudando a la identificación de potenciales fallas durante la ejecución de los procesos, permitiendo tomar las medidas de seguridad correspondientes. [4, 5, 8, 17]

La investigación de métodos que permitan el monitoreo de procesos es un tópico que ha ido adquiriendo más tracción debido a la gran disponibilidad de datos y avances tecnológicos en los últimos años tales como Industria 4.0 y Big Data. [2] Dentro de este paradigma, nos encontramos con la detección y diagnóstico de fallas, conjunto de técnicas para detectar, identificar y localizar anomalías, con tal de evitar (o minimizar) las consecuencias de estas fallas como también ayudar a las tareas de reparación. Entre estas técnicas, observamos tres grandes clasificaciones: los métodos basados en modelos, que usan modelos matemáticos para describir los procesos, los métodos basados en conocimientos, que generan sistemas considerando las reglas dadas por un experto, y los métodos basados en datos, que utilizan datasets para comprender y predecir el estado de los procesos, donde este último ha ido adquiriendo una mayor relevancia gracias al panorama actual y el menor grado de conocimiento requerido para su uso en comparación a los otros tipos de métodos, facilitando su aplicación. [6, 10, 13, 16, 18]

Entre las múltiples propuestas que han surgido sobre métodos basados en datos, con frecuencia se ve el uso de Machine Learning, especialmente de métodos no supervisados, ya que resultan una alternativa ideal ante los grandes volúmenes de datos disponibles, que usualmente carecen de etiquetas o referencias en bases de datos. En contraste, esto limita severamente la aplicación de métodos supervisados, incentivando aún más la exploración e investigación sobre métodos no supervisados (o semi-supervisados). [11, 14, 15]

El trabajo a presentar cumple con los objetivos n° 7 (sobre Energía y Desarrollo Sostenible) y n° 9 (sobre Industria, innovación e infraestructuras) de los Objetivos de Desarrollo Sostenible (ODS) [3], debido al uso de un dataset sobre paneles solares y un enfoque en técnicas de monitoreo de procesos dentro de un contexto industrial.

2. Objetivos

2.1. Objetivo General

Evaluar métodos de aprendizaje estadístico aplicados en el monitoreo de un proceso de conversión de energía en el contexto de industria 4.0 a través del diseño y ejecución de experimentos para la obtención de indicadores de desempeños (métricas).

2.2. Objetivos Específicos

- Realizar un análisis exploratorio sobre la base de datos a utilizar en el caso de estudio, para comprender la operación del proceso y la naturaleza de sus mediciones. Como resultados de este objetivo, se obtuvo un análisis estadístico del dataset (Tabla 2) en conjunto a una serie de gráficos (Figuras 3, 6 y Anexos).

- Investigar e implementar métodos de monitoreo de procesos seleccionados en conjunto con los cálculos de las métricas correspondientes, como se muestra (a rasgos generales) en la figura 1.

- Evaluar el rendimiento de los métodos de monitoreo implementados, a través del cálculo de métricas de desempeño, junto con el diseño y ejecución de experimentos para entrenar estructuras de clasificación no supervisadas. Los resultados obtenidos se ven resumidos en tablas (Tablas 4 a 11), mientras que el proceso que se utilizó para obtener estos resultados se aprecia en la figura 9.

2.3. Metodología

Entre los preparativos iniciales para la ejecución de esta memoria se encuentra la realización del análisis exploratorio de datos y correspondiente preprocesado del dataset a utilizar, de manera simultánea también se realiza la búsqueda de códigos que sirvan de ejemplos (principalmente a través de la plataforma GitHub), como también de librerías que faciliten la implementación de los métodos y métricas a utilizar para este trabajo.

Se implementaron un total de 4 métodos (PCA, PLS, KPCA y KPLS), en conjunto con el cálculo de 5 métricas de desempeño (T^2, SPE, Q^2, F1-Score

y Accuracy). Todos los elementos recién mencionados serán explicados en mayor detalle en la sección 4.2.

Con el preprocesamiento del dataset y la implementación de los métodos ya realizada, su aplicación será realizada en dos fases: En la primera se usa un set de entrenamiento y un set de testeo (generados a partir del dataset preprocesado) para obtener una serie de resultados iniciales que servirán como caso base para la segunda fase, la cual consistirá en la aplicación de una serie de anomalías generadas artificialmente y aplicadas sobre el set de testeo, para obtener resultados que permitan generar una comparación entre los diferentes métodos con la ayuda de las métricas anteriormente mencionadas.

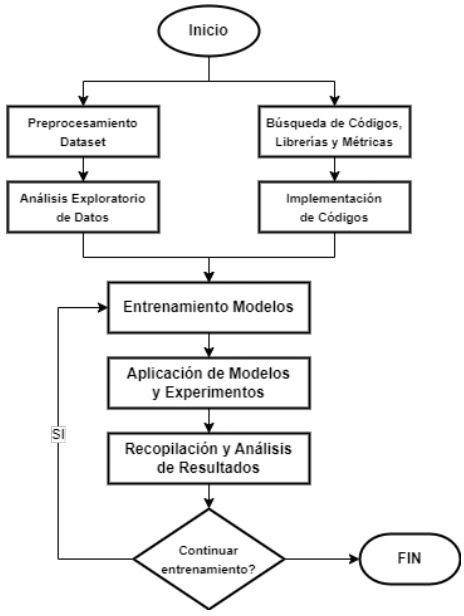


Figura 1: Diagrama de Flujo de la Metodología.

3. Marco Teórico

3.1. Industria 4.0

Industria 4.0 es el nombre utilizado para referirse a la cuarta revolución industrial, caracterizada como una optimización de la digitalización y automatización de diversos procesos que ocurrió durante la tercera revolución industrial (Industria 3.0) con el objetivo de descentralizar la producción a través de compartir diferentes establecimientos de producción, en lo que sería una red industrial conectada globalmente, permitiendo entregar productos o servicios con un mayor grado de personalización, cuya eficiencia además sería mejorada a través del uso de sistemas de análisis de datos. [2]

3.2. Big Data

Big Data es el término utilizado para referirse a los grandes volúmenes de datos que son imposibles de manejar con métodos y bases de datos tradicionales, como también a la capacidad de trabajar con estas grandes cantidades de datos. Los avances tecnológicos asociados con la Industria 4.0 y el IoT (Internet de las cosas) han llevado a la evolución de Big Data en tres grandes aspectos generales: el volumen de los datos producidos, la velocidad con la que se trabaja con estos datos (generación, análisis y respuesta) y la variedad de estos datos, ya sea con respecto a su origen o el tipo de dato generado, todo esto gracias al gran número de dispositivos y sensores presentes a día de hoy. [12]

3.3. Monitoreo de Procesos

El monitoreo de procesos abarca todo lo asociado al trabajo de observar en tiempo real un proceso con tal de verificar su estado actual, asegurando su correcta ejecución, convirtiéndolo en un aspecto de particular relevancia para el sector industrial con tal de mantener su productividad, una labor en constante cambio debido a la evolución tecnológica (Industria 4.0, Big Data) y las diversas propuestas que surgen gracias a esto, como se puede apreciar en la Figura 2. En la actualidad, el monitoreo de procesos dispone de una gran cantidad de información disponible gracias a la existencia de cada vez mejores sensores y sistemas asociados a estos procesos. [5]

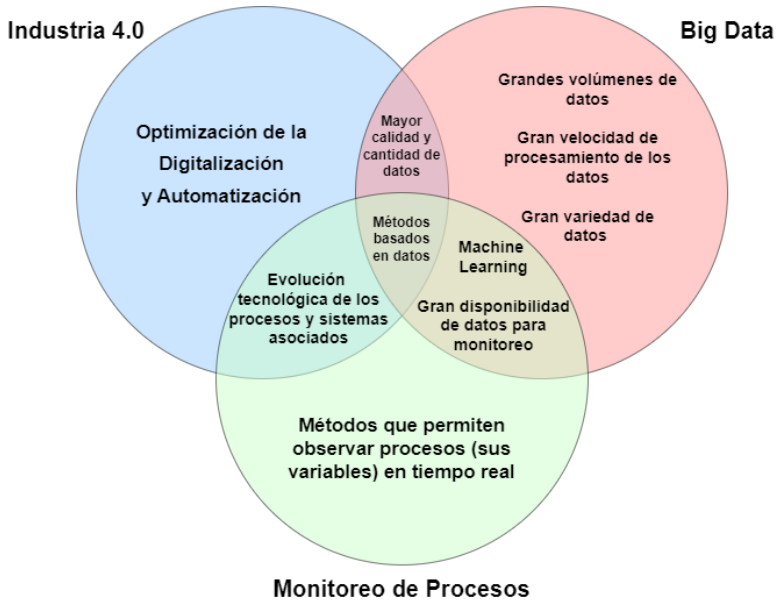


Figura 2: Relación entre Industria 4.0, Big Data y Monitoreo de Procesos.

3.4. Detección y Diagnóstico de Fallas

La detección y diagnóstico de fallas corresponde al conjunto de técnicas diseñadas para detectar, identificar, localizar y diagnosticar fallas a través del análisis cuantitativo y cualitativo de los datos históricos del proceso, con el objetivo de evitar (o minimizar) el efecto que estas fallas puedan tener como también el acelerar la reparación (o ajustes) correspondientes, siendo por ello una parte fundamental del monitoreo de procesos.[18] Los conjuntos de técnicas pertenecientes a esta área son clasificados (a grandes rasgos) en tres grandes clasificaciones:

3.4.1. Métodos basados en modelos

Este tipo de método se caracteriza por establecer modelos matemáticos que permitan describir un proceso de manera detallada, pero la tarea de establecer un modelo puede resultar ser muy compleja de realizar mientras mayor sea la envergadura del proceso que se esta tratando de definir, sin mencionar que los cálculos requeridos para ejecutar estos modelos matemáticos pueden ser imposibles de realizar en tiempo real en la práctica. [5, 18]

3.4.2. Métodos basados en conocimientos

Estos se caracterizan por generar un sistema a través del criterio y experiencia proveniente de un experto, donde este ayuda a idear una serie de reglas que el sistema debe cumplir o tener en cuenta para su funcionamiento. La dificultad para aplicar este tipo radica en la creación de un conjunto de reglas lo suficiente robusta para un sistema, sin mencionar que encontrar a un experto puede ser particularmente complejo, especialmente mientras más novedoso sea el proceso. [5, 18]

3.4.3. Métodos basados en datos

Los métodos basados en datos solo requieren de la existencia de datos disponibles para su aplicación, en conjunto a un nivel de conocimientos mucho menor con respecto al proceso en comparación a los otros dos métodos recién mencionados, lo que lo convierte en una alternativa ideal tomando en consideración el panorama actual (gran cantidad de datos disponibles), incentivando la investigación alrededor de este tipo de métodos (ya sea de manera individual o a través de métodos híbridos). [5, 18]

3.5. Aprendizaje Estadístico

El aprendizaje estadístico corresponde al conjunto de métodos que combinan principios de estadísticas y machine learning (aprendizaje supervisado o no supervisado). Dentro del aprendizaje estadístico nos encontramos con los métodos de análisis multivariable, que son capaces de analizar múltiples variables y las relaciones entre estas, permitiendo encontrar patrones y hacer predicciones. [1, 7]

3.6. Función Kernel

Herramienta matemática que traduce los datos usados como entrada (input) a un espacio de mayor dimensionalidad, permitiendo proyectar comportamientos no lineales presentes en estos datos a través de productos escalares, de manera que este tipo de comportamiento pueda ser detectado por algoritmos lineales, extendiendo el alcance y capacidad de estos. [1]

3.6.1. Kernel Gaussiano

También conocido como Kernel RBF (Radial Basis Function), este corresponde a un tipo de función kernel caracterizado por ser isotrópico (solo depende de la distancia euclidiana entre dos puntos) y estacionario (depende de la diferencia entre dos puntos, independiente de su posición absoluta). También posee una gran capacidad de interpolación (capacidad de estimar valores dentro de rangos conocidos) siempre y cuando el ancho de kernel no sea muy grande, en contraste con su capacidad de extrapolación (estimación de valores fuera de rangos conocidos) que es limitada. [1]

3.6.2. Kernel Sigmoide

También referido como un kernel de perceptrón multicapa, este tipo de función kernel es condicionalmente positivo definido, es decir que su matriz de Gram siempre será simétrica y con valores propios positivos bajo las condiciones correctas. Fuera de estas condiciones, el kernel puede presentar aproximaciones incorrectas o una capacidad muy baja de interpolación. [1]

3.6.3. Kernel Polinomial

Tipo de función kernel que presenta un comportamiento no estacionario y con una mejor capacidad para manejar la no linealidad presente en los datos (entrada y/o salida). Ofrece una mayor capacidad de extrapolación cuando se utilizan menores grados del polinomio, a cambio de una interpolación limitada. [1]

4. Desarrollo

4.1. Caso Real y Dataset

El dataset utilizado para el trabajo en esta memoria consiste en una serie de mediciones obtenidas *in situ* de variables meteorológicas y eléctricas a través de un sensor "Lufft WS502-UMB" durante un periodo de 4 años (Marzo 2019 – Febrero 2023) con respecto al funcionamiento de paneles solares ubicados en el edificio tecnológico mecánico perteneciente a la facultad de ingeniería.

El dataset contiene un total de 8 variables (especificadas en la tabla 1), cuyas mediciones fueron realizadas cada minuto, resultando en un dataset con más de 2 millones de entradas.

Originalmente estos datos se encontraban distribuidos en una serie de archivos CSV (un total de 8 archivos, uno por cada variable) conteniendo las diferentes mediciones obtenidas en conjunto a la fecha y hora en que se obtuvo la medición, los cuales fueron unificados en un único archivo csv para trabajar con el dataset en posteriores etapas.

Variable	**Descripción**
Time	Fecha y hora en que se realizó la medición.
Humidity	Mediciones de humedad relativa. Medido en porcentajes [%].
Irradiance	Mediciones de irradiación horizontal global (GHI). Medido en Watts partido por metro cuadrado $[W/m^2]$
Power	Potencia de salida medida en Watts $[W]$.
Pressure	Presión atmosferica medida en Bares $[bar]$.
Temperature	Temperatura del ambiente medida en grados Celsius $[°C]$.
Voltage DC	Mediciones del Voltaje DC medidos en Voltios $[V]$.
Voltage AC	Mediciones del Voltaje AC medidos en Voltios $[V]$.
Windspeed	Velocidad del viento medida en metro partido segundo $[m/s]$.

Tabla 1: Resumen del Dataset de Paneles Fotovoltaicos.

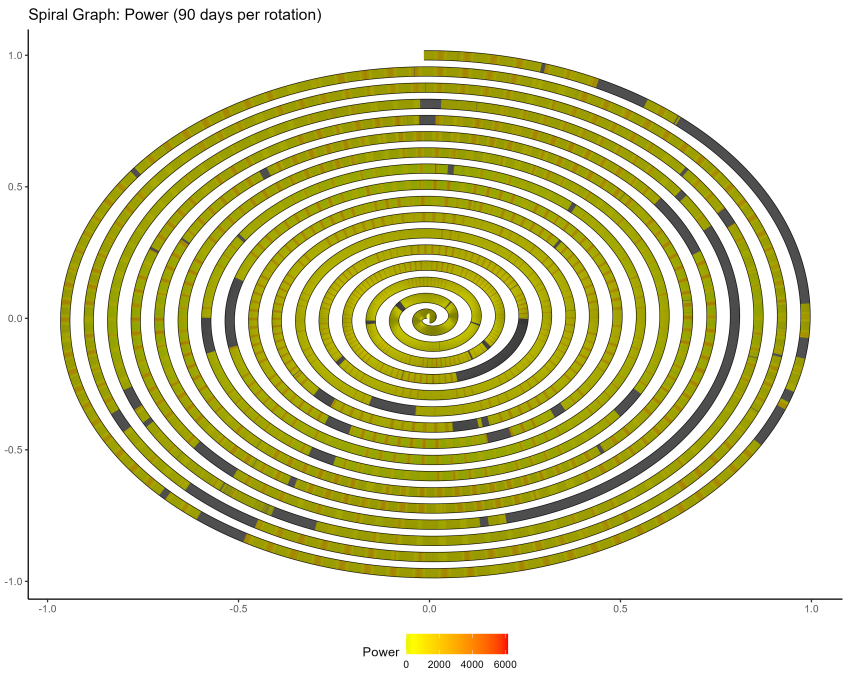


Figura 3: Gráfico en espiral del Dataset completo. (Valores Nulos en Gris)

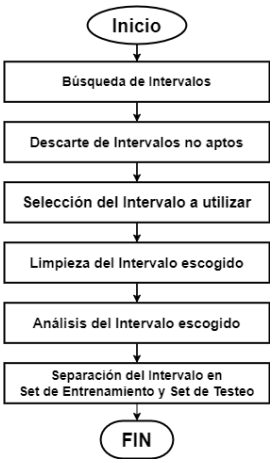


Figura 4: Diagrama de Flujo del Preprocesado del Dataset.

4.1.1. Reducción del Dataset

Como se menciono anteriormente, el dataset cuenta con más de 2 millones de entradas, por lo tanto se hace necesario reducir su tamaño escogiendo un intervalo más pequeño, con los objetivos de tener un dataset con un periodo de tiempo más manejable (en términos de recursos), como también para reducir el trabajo requerido para la limpieza del dataset al evitar periodos con grandes aglomeraciones de valores nulos como los que se pueden ver en la figura 3, en donde se observan los valores de la variable "Power", una de las variables con la mayor cantidad de valores nulos presentes, y que con suma frecuencia coincide con la presencia de valores nulos de otras variables.

El intervalo de tiempo debe cumplir con las siguientes condiciones: (1) Debe tener una longitud aproximada de 2 a 3 meses y (2) La presencia de valores nulos, los que usualmente son originados por errores de medición, fallos en el cálculo y/o transmisión de los datos o por periodos en donde los paneles solares o sus sensores estuvieron deshabilitados por circunstancias particulares, debe no ser mayor a un 5 % del total de datos presentes en el intervalo. Para ello se realizó un preprocesamiento sobre el dataset para hallar este intervalo, el cual se puede apreciar en la figura 4.

Para encontrar intervalos que cumplan con las condiciones recién mencionadas, se revisó de principio a fin el dataset en intervalos de 3 meses y de 2 meses, y en cada intervalo a revisar se calculaba el porcentaje de valores nulos (representados por NaN) presentes en el intervalo. Para efectos de este cálculo, se considera como valor nulo cualquier medición individual (independiente de la variable o marca de tiempo) que no registró un valor válido en el momento en que se realizó la medición, y por lo tanto entregó un NaN (Not a Number) en su lugar. Esto no tiene ningún efecto sobre la entrada en la marca de tiempo correspondiente, independiente de la cantidad de valores nulos que pueda contener, ya que el objetivo de este proceso es tener la mayor cantidad de mediciones validas dentro del intervalo. El porcentaje de valores nulos se obtuvo a través de la ecuación 1 presentada a continuación:

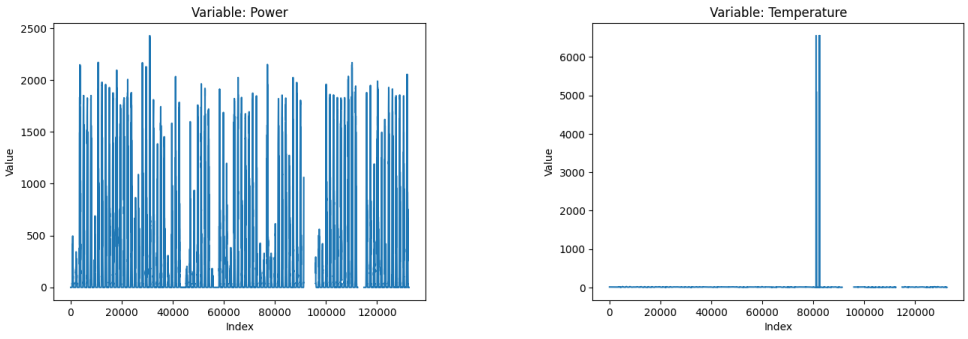
$$\%\ \text{Valores Nulos} = \left(\frac{\text{Cantidad de Valores Nulos (NaN)}}{\text{Cantidad Total de datos del intervalo}}\right) \times 100 \quad (1)$$

La cantidad de valores nulos fue obtenida a través del siguiente comando: *df.isnull().sum().sum()*, con *df* siendo el dataframe que alberga el intervalo

extraído durante la revisión, la función *isnull()* indica (a través de un booleano) si el dato es nulo (Verdadero) o no (Falso), y finalmente con el primer *sum()* se obtiene la cantidad de valores nulos por variable, mientras que el segundo *sum()* obtiene la sumatoria total del dataset. La cantidad total de datos presentes es obtenida como el resultado de la multiplicación entre la cantidad de filas que contiene el intervalo por 8, que es el número de columnas con datos numéricos (variables). Finalmente, se realiza la multiplicación de estos valores por 100 para convertir el resultado en un valor porcentual.

Tras este proceso inicial, se obtuvieron un total de 31 posibles intervalos de 3 meses y 27 posibles intervalos de 2 meses. Para filtrar esta cantidad de potenciales intervalos, estos fueron revisados más detenidamente a través de gráficos de línea (lineplots), histogramas y valores estadísticos tales como media, mediana, máximo y mínimo, para cada columna del dataset.

Durante la revisión, se aprecian dos situaciones en los gráficos de línea que facilitaron el proceso de eliminación de intervalos: la primera situación (y la más frecuente) consiste en que varios de los intervalos presentan gráficos con porciones vacías que interrumpen la continuidad de estos tal como se muestra en la figura 5a, dando a entender que existe agrupaciones de valores nulos contiguos notoriamente grandes dentro del intervalo, la segunda situación es la presencia de valores anormales (sumamente altos o bajos) particularmente observables en las variables de temperatura y presión, lo que entregaba gráficos distorsionados y que no aportaban información precisa sobre el intervalo, apreciable en la figura 5b.



(a) Gráfico interrumpido.

(b) Valores anormales.

Figura 5: Ejemplos de errores en los datos.

Finalmente, esta revisión concluye con un total de 4 posibles intervalos de 3 meses más 8 posibles intervalos de 2 meses, en donde se les dio prioridad a los intervalos de 3 meses debido a que albergaban más datos. Particularmente el intervalo escogido para trabajar es el intervalo que comprende desde Marzo hasta Mayo del año 2019, ya que posee el menor porcentaje de valores nulos presentes (0.156 %) dentro de los intervalos de 3 meses.

4.1.2. Limpieza del Dataset

Para la limpieza de los valores nulos (NaN) presentes en el intervalo escogido, se optó por realizar una imputación simple de valores (reemplazar NaN con media o mediana) para esto es importante observar el tipo de distribución que poseen las diferentes variables presentes en el dataset, con tal de escoger la opción más adecuada a cada caso. Para ello, se utilizaron histogramas (con 20 bins) para observar estas distribuciones en conjunto de observar los valores de media y mediana, de lo cual se llego a las siguientes conclusiones: la humedad, temperatura y presión serán rellenados con la media, poder, irradiación y velocidad del viento serán rellenados con la mediana, y finalmente ambos voltajes serán rellenados con la moda, debido a que estos presentaban una distribución bimodal, y por lo tanto la imputación de valores utilizando la media o la mediana no sería la alternativa más correcta. Por otro lado, los valores negativos (resultantes por errores de medición) fueron simplemente reemplazados por cero. Los resultados de la limpieza del dataset pueden ser observados en la figura 6.

4.1.3. Análisis del Dataset

Para proseguir con el análisis exploratorio de datos, se generaron una serie de gráficos por característica, entre ellos se encuentran algunos de los gráficos ya mencionados como los lineplots para ver la progresión y continuidad de los datos, histogramas para ver la distribución de las variables en conjunto a valores de estadísticos tales como la media y mediana.

En adición a los gráficos recién mencionados también se generaron varios heatmaps para ver la relación entre las variables del dataset. Entre estos se tiene un heatmap básico, un heatmap de información mutua y dos heatmaps de transferencia de entropía, uno de estos con los datos naturales del dataset y el otro con los datos después de aplicar un escalado logarítmico sobre estos.

Los cálculos de estos últimos tres heatmaps fueron obtenidos a través de la herramienta JIDT (Java Information Dynamics Toolkit), y fueron realizados para observar la relación entre la potencia con el resto de las variables. Para acelerar la obtención de estos resultados se paralelizó el procedimiento para poder realizar dos series de cálculos simultáneamente. Todos los gráficos recién mencionados se encuentran en los anexos de este documento.

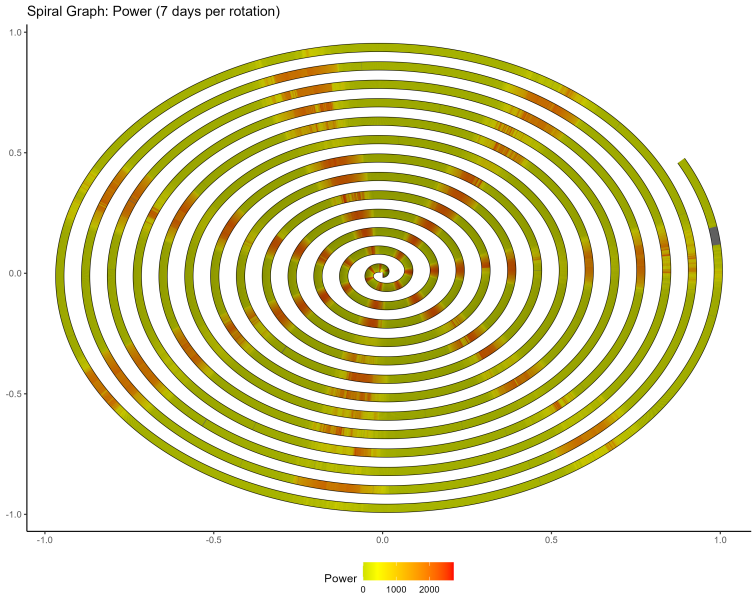
4.1.4. Intervalo Resultante

Como se mencionó anteriormente, el intervalo seleccionado corresponde al periodo comprendido entre los meses de Marzo 2019 hasta Mayo 2019 (3 meses). Un análisis estadístico de este intervalo se puede ver en la tabla 2.

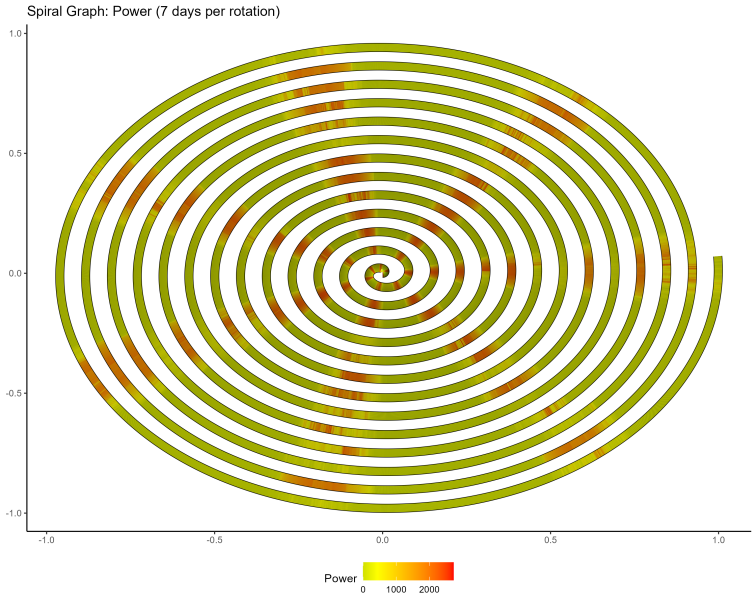
El dataset cuenta con un total de 131154 filas de datos, el cual fue separado en dos porciones: un set de entrenamiento (training set) usando los primeros dos meses (Marzo y Abril) con un total de 86865 entradas de datos, mientras que la otra porción corresponde a los datos de Mayo (44289 entradas de datos) que servirán como set de testeo (testing set) para el trabajo con los modelos de monitoreo de procesos.

Variable	**Media**	**Varianza**	**Skewness**	**Kurtosis**	**Rango**
Humidity	81.97	289.63	-1.045	0.149	[19.3, 100.0]
Irradiance	154.81	64451.98	1.58	1.245	[0.0, 1301.164]
Power	431.85	489865.79	1.45	0.616	[0.0, 2731.273]
Pressure	1.014	0.000011	-0.101	0.202	[1.004, 1.025]
Temperature	13.466	19.511	0.460	0.302	[1.9, 30.3]
Voltage DC	134.967	20438.985	0.135	-1.948	[0.0, 361.1]
Voltage AC	105.810	12389.598	0.102	-1.987	[0.0, 231.3]
Windspeed	0.758	0.62	1.648	3.433	[0.0, 6.87]

Tabla 2: Análisis estadístico del intervalo seleccionado.



(a) Antes del Preprocesado. (Valores Nulos en Gris)



(b) Después del Preprocesado.

Figura 6: Gráficos en espiral del intervalo seleccionado.

4.2. Modelos de Monitoreo de Procesos

4.2.1. Modelos implementados

En esta sección se presentan los 4 modelos implementados para esta memoria (PCA, PLS, KPCA y KPLS) en conjunto a una explicación básica de su funcionamiento, finalizando con la mención de las variaciones ejecutadas de estos modelos y sus respectivas configuraciones. Todos los modelos fueron producidos en lenguaje Python, apoyándose en las implementaciones y herramientas proporcionadas por la librería *sklearn*.

Principal Component Analysis

Principal Component Analysis (PCA) corresponde a una técnica de modelado estadístico multivariable, específicamente de reducción de dimensionalidad, en donde un espacio de dimensión $\boldsymbol{m}$ (conjunto de datos con $\boldsymbol{m}$ variables o características, usualmente referido como X) es reducido a un espacio de dimensión $\boldsymbol{l}$, con $\boldsymbol{l}$ siendo la cantidad de componentes principales a retener y cuyo valor es estrictamente menor a $\boldsymbol{m}$.

La reducción de dimensionalidad se logra a través de producir un conjunto de combinaciones lineales ortogonales que permita descomponer X, en donde se selecciona el subconjunto de estas combinaciones (con respecto al valor de $\boldsymbol{l}$, es decir el número de componentes) que permita capturar y resumir de mejor manera la variabilidad de los datos.

$$X = TP^T + E \tag{2}$$

En la ecuación 2 se muestra la descomposición de X en el subconjunto de combinaciones lineales representado por TP^T, en donde T representa la matriz de puntuaciones y P representa la matriz de cargas, mientras que E corresponde a la matriz de residuales, cuya información proviene de interferencias o ruido detectado dentro de las mediciones contenidas en X. [9]

Partial Least Squares Regression

Partial Least Squares (PLS) corresponde a una técnica de modelado estadístico multivariable, específicamente a un modelo de regresión lineal que trabaja buscando la relación estadística entre dos espacios de observación: X e Y, en donde X corresponde a las variables independientes o entrada

(input), mientras que Y corresponde a las variables dependientes o salida (output).

De manera similar a PCA, PLS busca encontrar una serie de combinaciones lineales que permitan descomponer X e Y capturando la mayor variabilidad de los datos posible (con respecto al número de componentes $\boldsymbol{l}$ definido) como se muestra en las ecuaciones 3 y 4.

$$X = TP^T + E \tag{3}$$

$$Y = UQ^T + F \tag{4}$$

En donde T y U representan las matrices de puntuaciones (para entrada y salida respectivamente), mientras P y Q corresponden a las matrices de cargas y con E y F siendo las matrices residuales para X e Y respectivamente.

Aparte de encontrar un conjunto de combinaciones lineales para descomponer X e Y, como se mencionó anteriormente, PLS también busca encontrar una relación entre estos dos espacios de observación en la forma de una combinación lineal con el propósito de poder predecir los valores de Y en base a X, como se aprecia en la ecuación 5:

$$Y = XB + F \tag{5}$$

En donde Y es el resultado de la multiplicación entre X y B, que corresponde al coeficiente de regresión, todo esto sumado F, que representa los valores residuales del modelo. [9]

Kernel PCA y Kernel PLS

En el caso de los métodos KPCA y KPLS, el funcionamiento general de estos métodos se mantiene intacto con respecto a su respectiva versión original. La diferencia radica en que en vez de utilizar el dataset de manera directa al aplicar el método, ahora se utiliza una herramienta matemática conocida como kernel, el cual es generado a partir del dataset. Un kernel corresponde a una matriz en donde sus valores representan la similitud o relación entre dos datos (x_i, x_j), los cuales son obtenidos de lo que se denomina como función kernel $K(x_i, x_j)$, la cual cumple con la propiedad expresada en la ecuación 6.

$$K(x_i, x_j) = \phi(x_i)^\top \phi(x_j) \tag{6}$$

En esta ecuación (conocida como kernel trick) vemos que el resultado de una función kernel es equivalente al producto interno entre $\phi(x_i)^\top$ y $\phi(x_j)$, en donde $\phi(x)$ corresponde a una función de mapeo que transforma los valores de x a un espacio de mayor dimensión, en donde las relaciones no lineales presentes en los datos ahora se vuelven lineales, lo cual corresponde a la gran ventaja que trae consigo utilizar el kernel, ya que este expande las capacidades iniciales de los métodos PCA y PLS (que trabajan sobre relaciones lineales) para ser capaces de trabajar con las relaciones no lineales presentes en el dataset. [9]

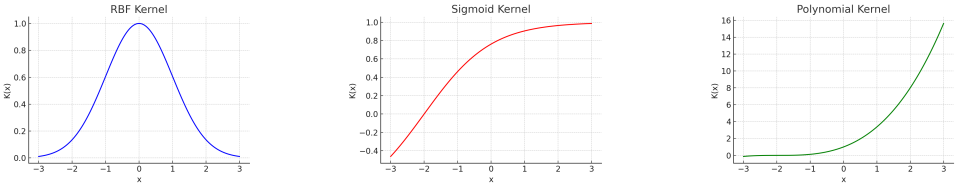
Existen varios tipos de funciones kernel, pero para efectos de este trabajo solo se utilizan tres: RBF (Radial Basis Function), sigmoide y polinomial. El kernel es generado utilizando la función *pairwise_kernels* de la librería *sklearn*, cuyas funciones kernel se muestran a continuación:

$$\text{RBF} := K(x_i, x_j) = \exp\left(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2\right) \tag{7}$$

$$\text{Sigmoid} := K(x_i, x_j) = \tanh\left(\gamma \langle x_i, x_j \rangle + \text{coef0}\right) \tag{8}$$

$$\text{Polynomial} := K(x_i, x_j) = \left(\gamma \langle x_i, x_j \rangle + \text{coef0}\right)^{\text{degree}} \tag{9}$$

En donde el parámetro gamma (γ) maneja cuanta influencia tienen los datos con respecto a otros datos según la distancia entre ellos (mayor sea el valor, menos influencia tienen los datos más lejanos), el parámetro coef0 corresponde a un valor fijo que ayuda a ajustar el impacto de los datos transformados (menor sea el valor, más dependiente es el kernel del producto interno entre x_i y x_j), y por último el degree ajusta el grado del polinomio, lo que afecta a la complejidad de la transformación, como también a la capacidad del modelo de capturar relaciones más complejas. Una representación gráfica de estas funciones puede verse a continuación en la figura 7.



(a) Kernel RBF. (b) Kernel Sigmoide. (c) Kernel Polinomial.

Figura 7: Funciones kernel utilizadas.

Configuraciones

Para efectos de esta memoria, se ejecutaron un total de 9 variaciones a lo largo de los 4 métodos anteriormente descritos: 2 versiones de PCA, referidas como PCA-01 y PCA-02, una versión de PLS y en el caso de KPCA y KPLS, por cada uno se ejecutaron tres versiones (una por cada kernel especificado anteriormente), en donde las configuraciones específicas para cada variación se pueden ver en la tabla 3 presentada a continuación.

Modelo	**n_components**	**gamma**	**coef0**	**degree**
PCA-01	5	-	-	-
PCA-02	0.99	-	-	-
PLS	4	-	-	-
KPCA (RBF)	30	0.04	-	-
KPCA (Sigmoid)	15	0.04	1	-
KPCA (Poly)	20	0.04	1	2
KPLS (RBF)	30	0.04	-	-
KPLS (Sigmoid)	15	0.04	1	-
KPLS (Poly)	20	0.04	1	2

Tabla 3: Resumen configuraciones ejecutadas.

Nota: En el caso de PCA-02, el comportamiento de este modelo consiste en retener tantos componentes principales como sea necesario para obtener una varianza en los datos equivalente al decimal especificado como número de componentes.

4.2.2. Métricas Utilizadas

Hotelling T^2

Hotelling T^2 (simplemente referido como T^2) es una métrica que permite observar la variabilidad capturada por el modelo y sus componentes. Con un limite de control (T^2_α) establecido, T^2 permite definir si una muestra es normal ($T^2 \leq T^2_\alpha$) o una anomalía ($T^2 > T^2_\alpha$).

T^2 es obtenido como resultado de la sumatoria de las puntuaciones (scores) al cuadrado dividido la varianza de cada componente retenido en el modelo.

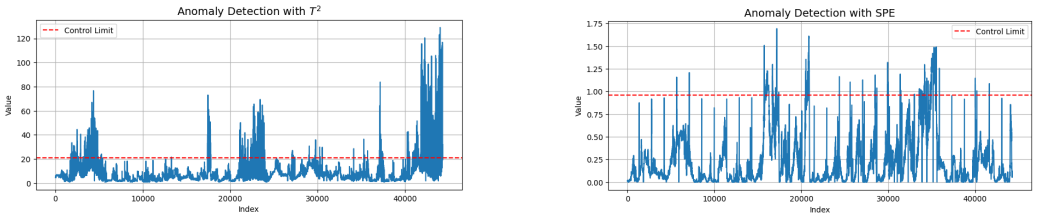
$$T^2 = \sum_{i=1}^{A} \frac{t_i^2}{\lambda_i} \tag{10}$$

Squared Prediction Error

Simplemente referido como SPE, a través esta métrica se observa la variabilidad no capturada por el modelo, es decir la variabilidad presente en el espacio residual. En conjunto con T^2, es una métrica que permite definir si una predicción es normal o una anomalía (outlier) cuando se supera el limite de confianza establecido, como se observa en la figura 8.

El cálculo de SPE consiste en la diferencia al cuadrado entre los valores reales (dataset) y la predicción hecha por el modelo, a esta diferencia usualmente se le conoce como valores residuales (o simplemente residuales).

$$\text{SPE} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \tag{11}$$



(a) Detección Anomalías con T^2.

(b) Detección Anomalías con SPE.

Figura 8: Ejemplos detección de anomalías con el uso de métricas.

Prediction Power

El poder predictivo (usualmente referido como Q^2) permite analizar cuan bien el modelo generaliza o se adapta a nuevos datos, en otras palabras es un indicador de la efectividad del modelo con respecto a la calidad de sus predicciones. El valor de esta métrica generalmente está entre 0 y 1, con valores cercanos a 1 indicando una mayor calidad en las predicciones realizadas. A continuación se presenta la fórmula que permite el cálculo de Q^2:

$$Q^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{12}$$

Donde el numerador corresponde a la sumatoria de los valores residuales. Mientras que el denominador corresponde a la sumatoria de los cuadrados de la diferencia entre los valores reales y su media.

F1-Score

F1-Score corresponde a la media harmonizada entre Precision, que representa la proporción de verdaderos positivos con respecto a los valores positivos obtenidos (verdaderos o falsos) e indica la capacidad del modelo para evitar clasificar valores que en realidad son negativos como positivos y Recall, que corresponde a la proporción de valores que fueron correctamente clasificados como verdaderos positivos con respecto a todos los valores positivos del dataset (verdaderos positivos + falsos negativos) e indica la habilidad del modelo de identificar todos los valores positivos.

$$\text{F1-Score} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{13}$$

En la ecuación 13 tenemos la fórmula para calcular la métrica F1-Score, en donde TP se refiere a los verdaderos positivos (True Positives), FP corresponde a los falsos positivos (False Positives) y FN son los falsos negativos (False Negatives).

Accuracy

Accuracy calcula cuan precisas son las predicciones en comparación a los valores reales, en donde entrega un valor que corresponde a la proporción de predicciones que son correctas (coinciden con el valor o clasificación real) con respecto al total de las predicciones realizadas.

A continuación en la ecuación 14 tenemos la fórmula para calcular Accuracy, en donde TP, TN, FP y FN son los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos respectivamente.

$$\text{Accuracy} = \frac{TP + FP}{TP + TN + FP + FN} \tag{14}$$

4.2.3. Estructura del Código

Los códigos producidos para cada método siguen una estructura general, en donde cada método corresponde a una clase que incluye las funciones: ***init***, ***fit***, ***predict***, ***compare_predictions*** y ***plot_model***. En el caso específico de los métodos kernelizados, se tiene adicionalmente una función llamada ***calculate_kernel***. A continuación, se otorgará una descripción breve de estas funciones, en conjunto a una explicación de los parámetros que reciben.

Init: Función que inicializa el modelo y sus variables internas. Entre los parámetros que el usuario puede configurar se tiene la cantidad de componentes a utilizar por el modelo con el párametro *n_components* (cuyo valor por defecto es 2), como también el porcentaje a utilizar para calcular los límites de control para las métricas T^2 y SPE (*conf_T2* y *conf_SPE* respectivamente, cuyos valores por defecto son 99). En el caso de los métodos kernelizados, el usuario también puede definir el kernel a utilizar ("rbf", "sigmoid" o "poly", con "rbf" siendo el valor por defecto), como también los parámetros relevantes para estos kernels (*gamma*, *coef* y *degree*, con sus valores por defecto siendo None, 1 y 3 respectivamente).

Calculate_kernel: Función exclusiva de los métodos kernelizados (KPCA y KPLS) que calcula el kernel del dataset usando la función *pairwise_kernels* con los parámetros correspondientes dependiendo del kernel seleccionado. En el caso de los métodos PLS y KPLS, solo se utilizan las variables independientes (X) para generar el kernel.

Fit: Función para ajustar el modelo utilizando el dataset de entrenamiento, calculando en el proceso los límites de control para las métricas T^2 y SPE. Esta función solo exige que se le proporcione el dataset correspondiente como parámetro (en PLS y KPLS, se pide el dataset en el formato de entrada y salida: X e Y). El funcionamiento general de la función se puede observar en el pseudo-código 1.

Pseudo-código 1: Función Fit

```
def fit(self, dataset):
    ValidateDataset();
    trainset = StandarizeDataset(dataset)

    calculate_kernel(trainset) # Solo para KPCA y KPLS.
    fit_model(trainset) # PLS: trainset = X, Y.

    scores, resids = get_scores_and_residuals()
    T2, SPE = calculate_metrics(scores, resids)
    calc_metrics_thresholds(T2, SPE)
    plot_variance() # Exclusivo para metodos PCA y KPCA.
```

Predict: Función que genera la predicción considerando el dataset de testeo provisto, durante el procedimiento la función calcula la métrica Q^2 como también genera un DataFrame con las predicciones catalogadas como "Normal" u "Outlier" (Anomalía) según las métricas T^2 y SPE. Ambos elementos (valor de Q^2 y DataFrame de resultados) son retornados al finalizar la función. Para su ejecución esta función exige dos condiciones: que se le proporcione el dataset (datasets X e Y en el caso de PLS/KPLS) y que la función *fit* haya sido ejecutada al menos una vez, es decir que el modelo haya sido ajustado. Su funcionamiento general se puede ver en el pseudo-código 2.

Pseudo-código 2: Función Predict

```
def predict(self, dataset):
    ValidateDataset();
    testset = StandarizeDataset(dataset)

    calculate_kernel(testset) # Solo para KPCA y KPLS.
    predict_model(testset) # PLS: testset = X, Y.

    scores, resids = get_scores_and_residuals()
    T2, SPE, Q2 = calculate_metrics(scores, resids)
    results = generate_results()

    return results, Q2
```

Compare_predictions: Función que permite comparar la predicción generada con respecto a un nuevo dataset con la predicción original (dataset de testeo), generando tres elementos en el proceso: (1) una serie de métricas con respecto a la nueva predicción en formato DataFrame, que incluye la cantidad de datos clasificados como 'Normal' y 'Anomalía', en conjunto a los valores de Q^2, F1-Score y Accuracy, (2) un gráfico comparativo mostrando los resultados obtenidos en ambos sobre la métrica seleccionada: T^2 o SPE (T^2, SPEx o SPEy para PLS y KPLS) y (3) una matriz de confusión con respecto a los resultados de la métrica seleccionada y el intervalo enfocado (si es que este es entregado como parámetro). Esta función retorna dos Dataframe (resultados de la nueva predicción y sus correspondientes métricas calculadas) más la matriz de confusión construida. El funcionamiento de esta función se puede observar en el pseudo-código 3.

Nota: En el caso de los métodos KPCA y KPLS, para ambas funciones (fit y predict) es posible solicitar que retorne el kernel calculado al entregar el parámetro "*get_kernel=True*", como también es posible entregar un kernel pre-calculado a través del parámetro *kernel*. De manera similar, en el caso de la función compare_predictions es posible evitar re-calcular una predicción y obtener los gráficos correspondientes a las otras métricas, entregándole los resultados y valor de Q^2 correspondientes de la predicción a través del parámetro *prediction* en una lista de la siguiente manera: [resultados, Q^2].

Plot_model: Función que permite graficar los resultados obtenidos por el modelo según la métrica especificada en el parámetro *metric*, en el caso de los métodos PCA y KPCA se tienen las siguientes opciones: "T2" o "SPE", mientras que los métodos PLS y KPLS pueden especificar las siguientes opciones: "T2", "SPEx" o "SPEy". Por defecto, los gráficos entregados por esta función son con respecto a la predicción original (dataset de testeo), pero es posible proporcionar otro set de resultados para graficar a través del parámetro *results*.

Pseudo-código 3: Función Compare_predictions

```
def compare_prediction(self, newData, mtr, focus_range):
    ValidateDataset(newData);

    # Generar nueva prediccion.
    newResults, newQ2 = self.predict(newData)
    summary = generate_summary(newResults, newQ2)

    # Generar graficos comparativos c/r a metrica (mtr).
    plot_model(results, mtr) # Prediccion original
    plot_model(newResults, mtr) # Nueva prediccion

    # Generar matriz de confusion.
    trueRes = results[mtr]
    predRes = newResults[mtr]
    cm = conf_matrix(trueRes, predRes, focus_range)

    return newResults, summary, cm
```

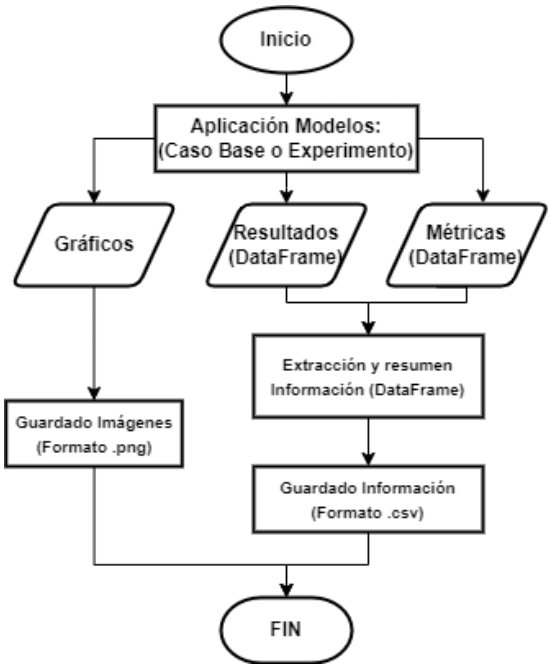


Figura 9: Diagrama de Flujo de metodología usada para obtener resultados.

4.3. Resultados: Caso Base

En esta sección se presentan los resultados obtenidos por los modelos durante su ejecución inicial, en donde se realiza el ajuste del modelo (función fit) usando el set de entrenamiento seguido de la obtención de la predicción inicial (función predict) usando el set de testeo. Los resultados presentados en esta y en la siguiente sección fueron obtenidos siguiendo la metodología descrita en la figura 9, y su ejecución fue realizada en el entorno TPU (gratuito) provisto por Google Colab, el cual posee 300 GB de memoria RAM en conjunto a 200 GB de espacio de almacenamiento.

A continuación, en las tablas 4 y 5 se presentan los resultados obtenidos por los modelos basados en PCA y los modelos basados en PLS respectivamente.

Modelo	Q^2	T^2	**SPE**	**Clasificación**
PCA-01	0.971257	41989 (2300)	42729 (1560)	40436 (3853)
PCA-02	0.996432	42091 (2198)	41211 (3078)	39031 (5258)
KPCA (RBF)	0.976766	42408 (1881)	41586 (2703)	41343 (2946)
KPCA (Sigm)	0.969598	41027 (3262)	41818 (2471)	40607 (3682)
KPCA (Poly)	0.980495	41843 (2446)	42301 (1988)	40149 (4140)

Tabla 4: Resultados de modelos PCA y KPCA

Modelo	Q^2	T^2	**SPE**	**Clasificación**
PLS	0.916576	43896 (393)	X: 40900 (3389) Y: 40219 (4070)	36734 (7555)
KPLS (RBF)	0.937712	44286 (3)	X: 41565 (2724) Y: 40475 (3814)	37905 (6384)
KPLS (Sigm)	0.932259	44103 (186)	X: 40425 (3864) Y: 40372 (3917)	36579 (7710)
KPLS (Poly)	0.930536	44056 (233)	X: 43974 (315) Y: 40225 (4064)	39869 (4420)

Tabla 5: Resultados de modelos PLS y KPLS

En las tablas se presentan cuatro columnas, en la primera se muestra el valor de la métrica Q^2, que representa la calidad de la predicción realizada, mientras que en las columnas restantes tenemos la clasificación realizada por las métricas T^2 y SPE en conjunto con la clasificación final hecha por el modelo considerando las ultimas dos métricas, en donde se muestra la

cantidad de datos calificados como 'Normal' en conjunto con la cantidad de datos calificados como 'Outlier' (Anomalía), con este último valor encerrado entre paréntesis. Para la clasificación final, si uno de los datos es clasificado como 'Outlier' ya sea por T^2 o SPE, la clasificación final sera 'Outlier'.
En el caso de los métodos PLS, la métrica SPE se calcula para ambos el input y el output (X e Y).

4.3.1. Análisis: Recursos vs Tiempo de Ejecución

En el caso de los métodos kernelizados, debido al tamaño del kernel generado (57 GB aprox) el proceso de ajuste del modelo, es decir la función fit resultó ser intensivo en recursos llegando a requerir 180+ GB de RAM de manera uniforme para los diferentes tiempos de ejecución de cada modelo, los cuales se pueden observar en la tabla a continuación:

Kernel	**Tiempo de Ejecución (KPCA)**	**Tiempo de Ejecución (KPLS)**
RBF	21 mins	48 mins
Sigmoide	22 mins	40 mins
Polinomial	18 mins	35 mins

Tabla 6: Tiempos de Ejecución (Aproximados)

Al observar los tiempos de ejecución obtenidos presentados en la tabla 6, podemos ver que aun si tienen el mismo gasto de memoria RAM, los modelos KPLS requieren aproximadamente el doble (incluso un poco más) de tiempo de ejecución en comparación al modelo KPCA que utiliza el mismo kernel bajo las mismas condiciones (cantidad de componentes retenidos, parámetros configurados).

4.4. Resultados: Aplicación Sintética de Anomalías

Para probar la efectividad de los modelos implementados, se aplicaron artificialmente una serie de anomalías de manera individual sobre el dataset de testeo, cada una de estas aplicada sobre una variable diferente. Todas las anomalías fueron aplicadas sobre el mismo rango, siendo este rango el comprendido entre los datos en la posición 5000 y 7000 del dataset mencionado.

4.4.1. Anomalía N° 1: Drift

Drift consiste en una desviación gradual (puede ser positiva o negativa) de las mediciones obtenidas en comparación a los valores reales durante un cierto periodo de tiempo. Este tipo de anomalía usualmente se presenta debido a factores tales como el envejecimiento y/o desgaste de los instrumentos de medición, condiciones ambientales, entre otros.

La anomalía fue generada siguiendo la ecuación 15, y esta fue aplicada sobre la variable de presión, observable en la Figura 10, mientras que los resultados obtenidos por los diferentes modelos se muestran en la Tabla 7.

$$x_{med}(k) = x_{real}(k) + d(k) \tag{15}$$

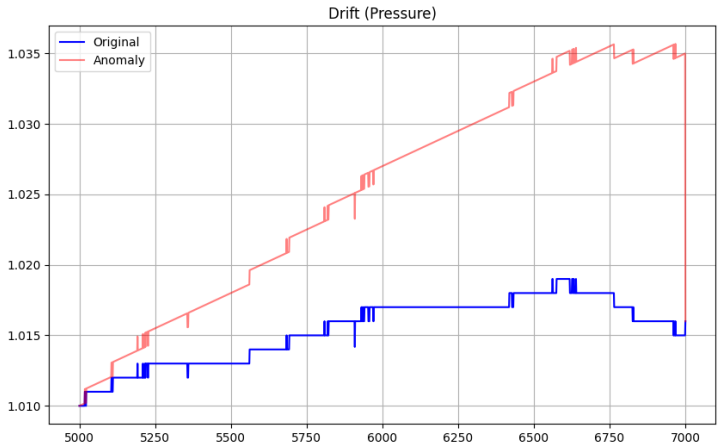


Figura 10: Anomalía Drift aplicado sobre Presión.

Modelo	Clasificación	Q^2	F1_Score	Accuracy
PCA-01	39003 (5286)	0.966128	0.981961	0.967644
PCA-02	37868 (6421)	0.996864	0.984824	0.973650
PLS	36400 (7889)	0.917499	0.994886	0.991555
KPCA (RBF)	40027 (4262)	0.961104	0.983827	0.970286
KPCA (Sigm)	39410 (4879)	0.955834	0.985041	0.972973
KPCA (Poly)	38814 (5475)	0.942420	0.982891	0.969496
KPLS (RBF)	36763 (7526)	0.922627	0.984036	0.973086
KPLS (Sigm)	35602 (8687)	0.932773	0.986271	0.977624
KPLS (Poly)	39016 (5273)	0.931925	0.988350	0.979250

Tabla 7: Resultados sobre Anomalía Drift

4.4.2. Anomalía N° 2: Noise

Noise corresponde a una serie de fluctuaciones aleatorias que son captadas por los sensores, lo que reduce la precisión de las mediciones realizadas. El ruido usualmente es provocado por problemas internos o distorsiones ambientales.

La anomalía fue generada siguiendo la ecuación 16, siendo aplicada sobre la variable de temperatura como se observa en la Figura 11. Los resultados obtenidos se pueden ver en la tabla 8.

$$x_{med}(k) = x_{real}(k) + n(k) \tag{16}$$

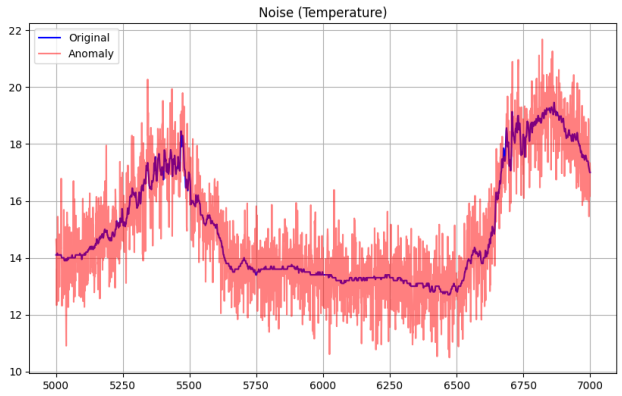


Figura 11: Anomalía Noise aplicado sobre Temperatura.

Modelo	Clasificación	Q^2	F1_Score	Accuracy
PCA-01	40399 (3890)	0.971146	0.999542	0.999165
PCA-02	39030 (5259)	0.996433	0.999962	0.999932
PLS	36732 (7557)	0.916577	0.999973	0.999955
KPCA (RBF)	41341 (2948)	0.976738	0.999976	0.999955
KPCA (Sigm)	40607 (3682)	0.969594	0.999975	0.999955
KPCA (Poly)	40137 (4152)	0.980465	0.999676	0.999413
KPLS (RBF)	37900 (6389)	0.937728	0.999802	0.999661
KPLS (Sigm)	36575 (7714)	0.932270	0.999891	0.999819
KPLS (Poly)	39869 (4420)	0.930545	0.999950	0.999910

Tabla 8: Resultados sobre Anomalía Noise

4.4.3. Anomalía N° 3: Packet Loss

Packet Loss ocurre cuando los paquetes conteniendo las mediciones realizadas se pierden durante la transmisión de estos (problemas de conexión), provocando una recolección incompleta o errónea de los datos, deteriorando fuertemente los procesos de monitoreo y control.

La anomalía fue generada siguiendo la ecuación 17, su aplicación fue realizada sobre la variable de Potencia como se observa a continuación en la Figura 12. Sus resultados se encuentran en la tabla 9.

$$x_{med}(k) = \begin{cases} x_{real}(k-\tau), & \text{con probabilidad } p, \\ 0, & \text{con probabilidad } 1-p, \end{cases} \tag{17}$$

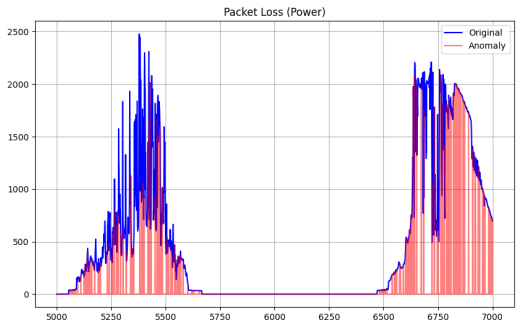


Figura 12: Anomalía Packet Loss aplicada sobre Potencia.

Modelo	**Clasificación**	Q^2	**F1_Score**	**Accuracy**
PCA-01	40078 (4211)	0.969108	0.995554	0.991917
PCA-02	38560 (5729)	0.994444	0.993414	0.988462
PLS	36212 (8077)	0.846946	0.992351	0.987401
KPCA (RBF)	41038 (3251)	0.975346	0.996298	0.993113
KPCA (Sigm)	40606 (3683)	0.969297	0.999988	0.999977
KPCA (Poly)	39696 (4593)	0.974887	0.993851	0.988914
KPLS (RBF)	37391 (6898)	0.866464	0.992749	0.987672
KPLS (Sigm)	36061 (8228)	0.862278	0.992401	0.987536
KPLS (Poly)	39363 (4926)	0.860282	0.993185	0.987807

Tabla 9: Resultados sobre Anomalía PacketLoss

4.4.4. Anomalía N° 4: Retained Values

Cuando ocurre una anomalía de Retained Values, las mediciones obtenidas por los sensores entregan un valor fijo durante un cierto periodo de tiempo, independiente de si el valor real ha cambiado durante ese periodo. Este tipo de anomalía usualmente se presenta debido a retrasos durante el procesamiento de los datos o por fallos de hardware.

La anomalía fue generada siguiendo la ecuación 18, y fue aplicada sobre la variable de Irradiación como se observa en la Figura 13, con los resultados en la tabla 10.

$$x_{med}(k + \Delta k) = x_{real}(k), \quad \Delta k \in [k_0, k_0 + T_{ret}] \tag{18}$$

Modelo	**Clasificación**	Q^2	**F1_Score**	**Accuracy**
PCA-01	40361 (3928)	0.970918	0.999072	0.998307
PCA-02	38934 (5355)	0.996098	0.997037	0.994784
PLS	36611 (7678)	0.906572	0.997450	0.995778
KPCA (RBF)	41267 (3022)	0.976528	0.999080	0.998284
KPCA (Sigm)	40607 (3682)	0.969585	1.000000	1.000000
KPCA (Poly)	40053 (4236)	0.979572	0.997581	0.995620
KPLS (RBF)	37741 (6548)	0.926839	0.997277	0.995349
KPLS (Sigm)	36419 (7870)	0.922011	0.996986	0.995033
KPLS (Poly)	39716 (4573)	0.920487	0.997273	0.995100

Tabla 10: Resultados sobre Anomalía Retained Values.

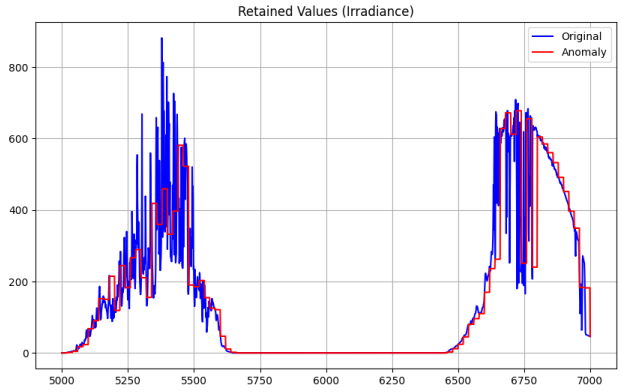


Figura 13: Anomalía Retained Values aplicado sobre Irradiación.

4.4.5. Anomalía N° 5: Packet Drop

A diferencia de Packet Loss, Packet Drop es una práctica intencional implementada para responder ante diferentes situaciones que se puedan presentar, algunos ejemplos serían el desbordamiento del buffer, errores de transmisión, problemas de red (debido a congestiones o restricciones), rechazo de paquetes por parte del cortafuegos, etc. Packet Drop también puede ser implementado como parte de una política de calidad de servicio (QoS Drop). De manera similar a Packet Loss, los paquetes descartados pueden generar una recolección de datos incompleta, o directamente generar lagunas (periodos sin datos), afectando la calidad de los procesos de monitoreo y control.

La anomalía fue generada siguiendo la ecuación 19 con $B(t)$ representando una distribución de Bernoulli, la cual es detallada en la ecuación 20.

$$x_{med}(k) = x_{real}(k) \times B(t) = \begin{cases} 0, & B(t) = 0 \\ x_{real}(k), & B(t) = 1 \end{cases} \tag{19}$$

$$B(t) \sim \text{Bernoulli}(1 - P_{drop}(t)) \tag{20}$$

Esta anomalía fue aplicada sobre la variable de Voltaje DC como se observa a continuación en la Figura 14, mientras que los resultados obtenidos se pueden observar en la tabla 11.

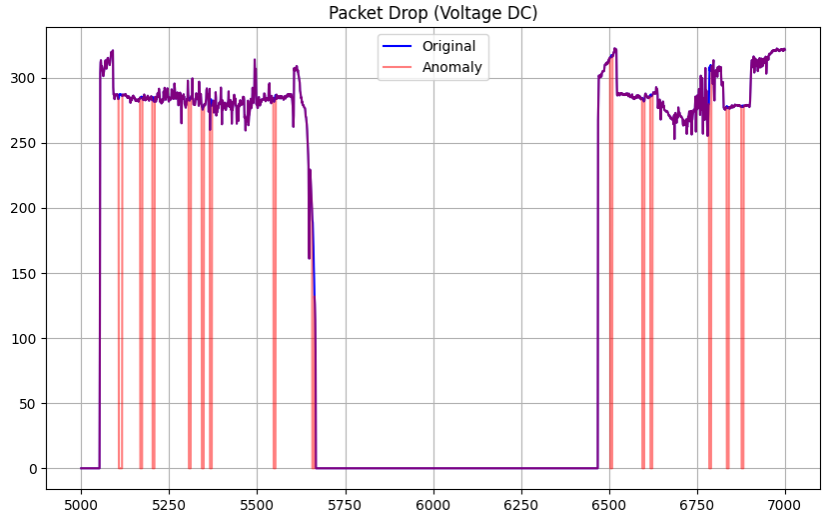


Figura 14: Anomalía Packet Drop aplicado sobre Voltaje DC.

Modelo	**Clasificación**	Q^2	**F1_Score**	**Accuracy**
PCA-01	40349 (3940)	0.970758	0.998923	0.998036
PCA-02	38963 (5326)	0.995953	0.999128	0.998465
PLS	36732 (7557)	0.916567	0.999973	0.999955
KPCA (RBF)	41268 (3021)	0.976658	0.999092	0.998307
KPCA (Sigmoid)	40584 (3705)	0.969423	0.999717	0.999481
KPCA (Poly)	40069 (4220)	0.979546	0.999003	0.998194
KPLS (RBF)	37840 (6449)	0.937509	0.999142	0.998532
KPLS (Sigmoid)	36540 (7749)	0.932103	0.999467	0.999119
KPLS (Poly)	39868 (4421)	0.930551	0.999987	0.999977

Tabla 11: Resultados sobre Anomalía Packet Drop.

5. Discusión y Conclusiones

En el presente documento, se explicó y detalló el trabajo realizado durante el transcurso de esta memoria lo que incluye: el preprocesado del dataset utilizado, los modelos de monitoreo de procesos implementados, su aplicación sobre el caso real más la ejecución de experimentos con anomalías artificiales, obteniendo una serie de resultados que permiten comparar el rendimiento de los métodos aplicados.

Con respecto a los resultados obtenidos, los modelos demuestran una buena capacidad para detectar las anomalías utilizadas en la gran mayoría de los casos, ya que se aprecia un aumento en la cantidad de anomalías detectadas al comparar los resultados de cada tipo de anomalía con los resultados base. Además, los altos valores de Q^2 (por sobre 0.85) indican que los modelos tienen una buena capacidad de adaptación ante nuevos datos sin caer en un sobreajuste. En particular, se destacan los resultados de los modelos PCA-02, KPCA (Polinomial) y KPCA (RBF) debido a la consistencia y precisión que demuestran al detectar las diferentes anomalías. Por otro lado, se tiene el caso del modelo KPLS (Polinomial) que fue incapaz de detectar la anomalía "Noise", y el caso del modelo KPCA (Sigmoide) que además de fallar en detectar la anomalía "Noise", también falló en detectar la anomalía "Retained Values".

Entre las limitaciones experimentadas, la principal radica en que el ambiente de Google Colab utilizado para las pruebas con los métodos kernelizados solo otorgaba entre 2 a 4 horas de ejecución diaria, ralentizando la etapa de pruebas y obtención de resultados, ya que transcurrido este periodo de tiempo, cualquier ejecución en progreso podría ser detenida por la desconexión forzada del ambiente, perdiendo cualquier resultado no guardado.

Por ello, una mejora relevante para los métodos implementados sería utilizar alguna implementación de aproximaciones de kernel para reducir los tiempos de ejecución como también el uso de RAM, especialmente dentro de un contexto con grandes volúmenes de datos como lo es el monitoreo de procesos.

6. Glosario

PCA : Principal Component Analysis.
PLS : Partial Least Squares.
KPCA : Kernel Principal Component Analysis.
KPLS : Kernel Partial Least Squares.
RBF : Radial Basis Function.
T^2 : Hotelling T^2.
SPE : Squared Prediction Error.
Q^2 : Prediction Power.
TP : True Positives.
TN : True Negatives.
FP : False Positives.
FN : False Negatives.

Referencias

[1] Joyce Chen Yen Ngu et al. «A comprehensive overview of the applications of kernel functions and data-driven models in regression and classification tasks in the context of software sensors». En: *Applied Soft Computing* 164 (2024), pág. 111975. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2024.111975. URL: https://www.sciencedirect.com/science/article/pii/S156849462400749X.

[2] Lian Duan y Li Da Xu. «Data Analytics in Industry 4.0: A Survey». En: *Information Systems Frontiers* (ago. de 2021). ISSN: 1572-9419. DOI: 10.1007/s10796-021-10190-0.

[3] M.J. Gamez. *Objetivos y metas de desarrollo sostenible.* Sep. de 2015. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[4] Cheng Ji y Wei Sun. «A Review on Data-Driven Process Monitoring Methods: Characterization and Mining of Industrial Data». En: *Processes* 10.2 (feb. de 2022), pág. 335. ISSN: 2227-9717. DOI: 10.3390/pr10020335.

[5] Qingchao Jiang, Xuefeng Yan y Biao Huang. «Review and Perspectives of Data-Driven Distributed Monitoring for Industrial Plant-Wide Processes». En: *Industrial & Engineering Chemistry Research* 58.29 (jul. de 2019), págs. 12899-12912. ISSN: 0888-5885. DOI: 10.1021/acs.iecr.9b02391.

[6] Ankur Kumar, Apratim Bhattacharya y Jesus Flores-Cerrillo. «Data-driven process monitoring and fault analysis of reformer units in hydrogen plants: Industrial application and perspectives». En: *Computers & Chemical Engineering* 136 (ene. de 2020), pág. 106756. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2020.106756.

[7] Yiqi Liu y Min Xie. «Rebooting data-driven soft-sensors in process industries: A review of kernel methods». En: *Journal of Process Control* 89 (2020), págs. 58-73. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2020.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0959152420300330.

[8] Lijia Luo et al. «Robust and sparse canonical correlation analysis for fault detection and diagnosis using training data with outliers». En: *Expert Systems with Applications* 236.29 (feb. de 2024), pág. 121434. ISSN: 09574174. DOI: 10.1016/j.eswa.2023.121434.

[9] Majdi Mansouri et al. *Data-Driven and Model-Based Methods for Fault Detection and Diagnosis*. Elsevier, feb. de 2020. ISBN: 9780128191644. URL: https://www.researchgate.net/publication/334448355_Data-Driven_and_Model-Based_Methods_for_Fault_Detection_and_Diagnosis.

[10] Afrânio Melo, Mauricio Melo Câmara y José Carlos Pinto. «Data-Driven Process Monitoring and Fault Diagnosis: A Comprehensive Survey». En: *Processes* 12.2 (feb. de 2024), pág. 251. ISSN: 2227-9717. DOI: 10.3390/pr12020251.

[11] Nimra Munir et al. «Machine Learning for Process Monitoring and Control of Hot-Melt Extrusion: Current State of the Art and Future Directions». En: *Pharmaceutics* 13.9 (sep. de 2021). ISSN: 1999-4923. DOI: 10.3390/pharmaceutics13091432. URL: https://www.mdpi.com/1999-4923/13/9/1432.

[12] Mustafa Bugran Ozcan, Batihan Konuk y Yildiz Merves Yesilcimen. «Big Data Analytics in Industry 4.0». En: *Industry 4.0: Technologies, Applications, and Challenges*. Springer Nature Singapore, dic. de 2023, págs. 171-199. ISBN: 978-981-19-2012-7. DOI: 10.1007/978-981-19-2012-7_8.

[13] You-jin Park, Shu-Kai S. Fan y Chia-Yu Hsu. «A Review on Fault Detection and Process Diagnostics in Industrial Processes». En: *Processes* 8.9 (sep. de 2020), pág. 1123. ISSN: 2227-9717. DOI: 10.3390/pr8091123.

[14] Rahul Rai et al. «Machine learning in manufacturing and industry 4.0 applications». En: *International Journal of Production Research* 59.16 (ago. de 2021), págs. 4773-4778. DOI: 10.1080/00207543.2021.1956675.

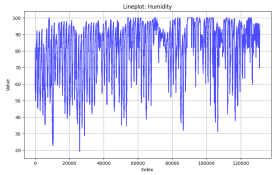
[15] Koosha Sharifani y Mahyar Amini. «Machine Learning and Deep Learning: A Review of Methods and Applications». En: *World Information Technology and Engineering Journal* 19.7 (mayo de 2023), págs. 3897-3904. URL: https://ssrn.com/abstract=4458723.

[16] Peng Tang, Kaixiang Peng y Jie Dong. «Nonlinear quality-related fault detection using combined deep variational information bottleneck and variational autoencoder». En: *ISA Transactions* 114 (ago. de 2021), págs. 444-454. ISSN: 00190578. DOI: 10.1016/j.isatra.2021.01.002.

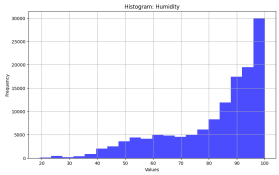
[17] Alexander Thebelt et al. «Maximizing information from chemical engineering data sets: Applications to machine learning». En: *Chemical Engineering Science* 252 (abr. de 2022), pág. 117469. ISSN: 00092509. DOI: 10.1016/j.ces.2022.117469.

[18] Jing Wang, Jinglin Zhou y Xiaolu Chen. *Data-Driven Fault Detection and Reasoning for Industrial Monitoring*. Springer Singapore, ene. de 2022. ISBN: 978-981-16-8043-4. DOI: 10.1007/978-981-16-8044-1.

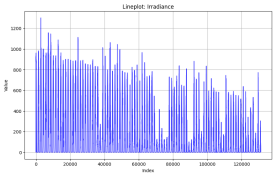
7. Anexos



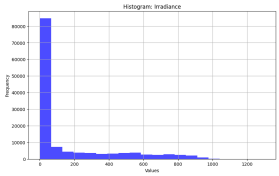
(a) Lineplot: Humedad.



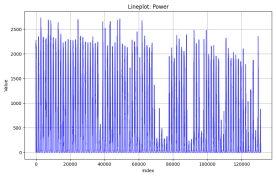
(b) Histograma: Humedad.



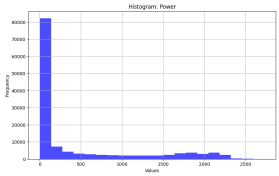
(c) Lineplot: Irradiación.



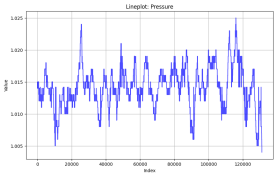
(d) Histograma: Irradiación.



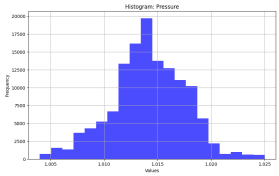
(e) Lineplot: Potencia.



(f) Histograma: Potencia.

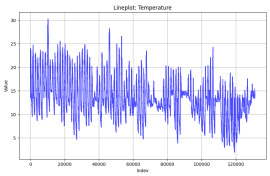


(g) Lineplot: Presión.

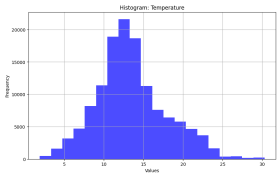


(h) Histograma: Presión.

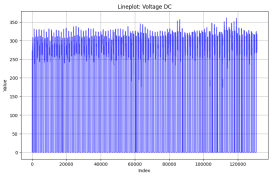
Figura 15: Gráficos Análisis Dataset (Parte 1).



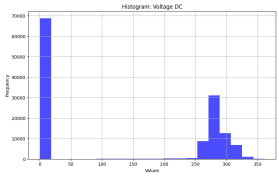
(i) Lineplot: Temperatura.



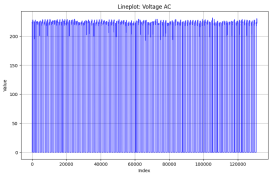
(j) Histograma: Temperatura.



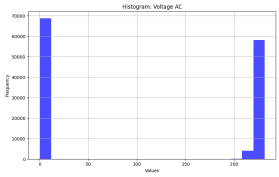
(k) Lineplot: Voltaje DC.



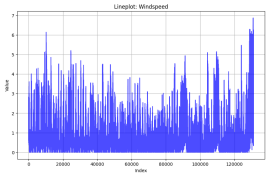
(l) Histograma: Voltaje DC.



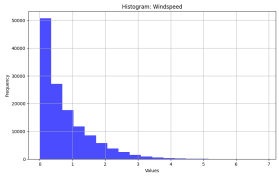
(m) Lineplot: Voltaje AC.



(n) Histograma: Voltaje AC.

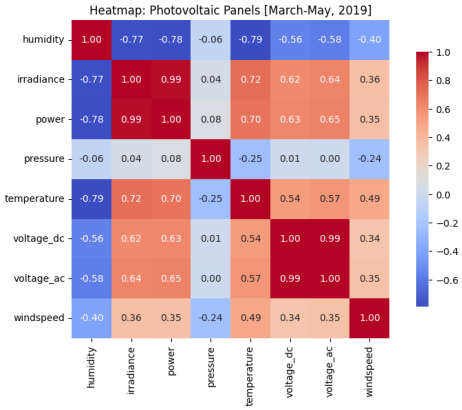


(o) Lineplot: Velocidad del viento.

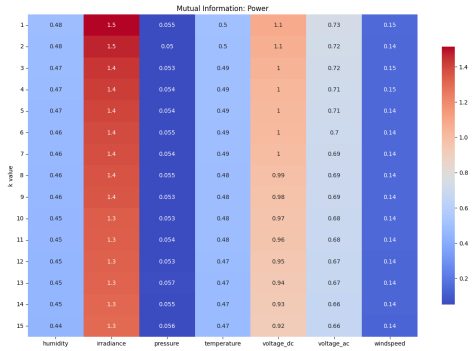


(p) Histograma: Velocidad del viento.

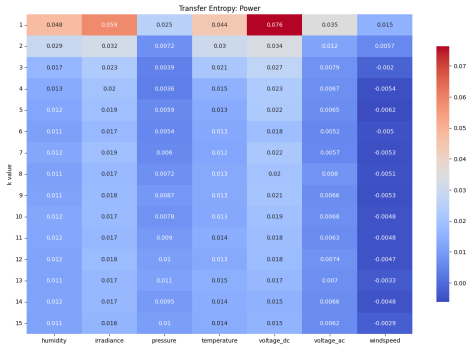
Figura 16: Gráficos Análisis Dataset (Parte 2).



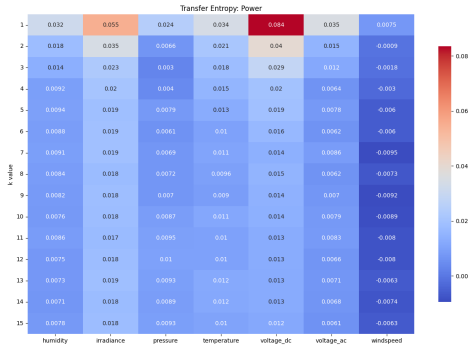
(a) Heatmap Básico



(b) Heatmap Información Mutua (Potencia).



(c) Heatmap Transferencia de Entropía (sin escalado).



(d) Heatmap Transferencia de Entropía (con escalado logarítmico).

Figura 17: Gráficos Análisis Dataset (Parte 3).