UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

TRABAJO DE FIN DE MÁSTER

ESTUDIO DE VALIDACIÓN, CORRECCIÓN Y ALMACENAMIENTO DE DATOS EN LA TELELECTURA DE CONSUMO DE AGUA DE TERRASSA

Autor: Antonio Gonzalez Cifuentes

Director: Alejandro Bachiller Matarranz
Co-director: Joan Van Eeckhout Alsinet

Titulación: Máster Universitario en Ingeniería de Sistemas Automáticos y Electrónica

Convocatoria: Primavera 2022

Agradecer, en primer lugar, a Joseba y Alejandro por adentrarme en el vasto mundo del tratamiento de datos y, junto a Joan, por aconsejarme y darme herramientas para poder llevar a cabo este proyecto que tanto he disfrutado.

A Iván, por haber compartido conmigo esta aventura académica y a Núria por haberme ayudado a conseguir el tiempo para culminarla satisfactoriamente.

Resumen

La red de distribución de agua de la ciudad de Terrassa contaba en 2021 con cerca de 100.000 contadores, de los cuales, menos del 1% disponía de telelectura.

Desde el año 2019, los organismos administrativos de la red llevan impulsado la instalación de telecontadores con la finalidad de dotar al sistema de una gestión más eficiente, sostenible y óptima. Con este objetivo se decide establecer un conjunto de métodos de tratamiento y validación de datos acorde a las necesidades de la red.

Este proyecto se centra en el estudio e implementación de estos métodos y su posterior almacenamiento. Para el desarrollo del mismo, el organismo municipal de gestión del agua de Terrassa ha facilitado información sobre los diversos telecontadores repartidos por la ciudad, así como de las telemedidas recogidas.

La propuesta metodológica en la validación de datos toma inspiración en la norma AENOR UNE500540, adaptada para su aplicación en la validación de los datos horarios de los telecontadores que, deben pasa un conjunto de tests de coherencia y fiabilidad para confirmarse como válidos. En caso contrario deben ser reconstruidos con alguno de los métodos propuestos en esta memoria.

Finalmente los datos serán importados a un servidor de bases de datos desde el cual se podrán realizar consultas para su posterior análisis.

Abstract

The water distribution network of the city of Terrassa had nearly 100,000 meters in 2021, of which less than 1% had remote reading.

Since 2019, the administrative bodies of the water network have promoted the installation of remote meters in order to provide the system with a more efficient, sustainable and optimal management. With this objective, it was decided to establish a set of data treatment and validation methods according to the needs of the network.

This project focuses on the study and implementation of these methods and their subsequent storage. For its development, the Terrassa municipal water management body has provided information on the various remote meters distributed throughout the city, as well as the remote measurements collected.

The methodological proposal for data validation is inspired by the AENOR UNE500540 standard, adapted for its application in the validation of hourly data from remote meters, which must pass a set of consistency and reliability tests to be confirmed as valid. Otherwise, they must be reconstructed with one of the methods proposed in this report.

Finally, the data will be imported to a database server from which queries can be made for later analysis.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Índice de Ecuaciones

Abreviaturas

Tabla 1.- *Tabla de Acrónimos*

Acrónimo	Significado
AR	Modelo AutoRegresivo
BBDD	Bases de Datos
CSV	Comma Separated Values
CV	Coeficiente de variación
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DQL	Data Query Language
ER	Entidad-Relación
ETL	Extract, Transform & Load
GUI	Graphical User Interface
IoT	Internet of Things
ML	Machine Learning
MM	Moving Mean
OAT	Observatorio de Aguas de Terrassa
RDBMS	Relational DataBase Management Systems
SQL	Structured Query Language
TAIGUA	Agua Municipal de Terrassa

1. Introducción al proyecto

El presente documento recoge la memoria técnica del Trabajo de Fin de Master en Ingeniería de Sistemas Automáticos y Electrónica Industrial (especialidad en Tecnologías de la Producción y Automatización Avanzadas) realizado por Antonio Gonzalez Cifuentes y co-tutorizado por Alejandro Bachiller Matarranz y Joan Van Eeckhart Alsient en el periodo que comprende desde enero hasta junio del presente año 2022.

El proyecto se centra en el tratamiento y almacenamiento automatizado de datos en la red de telecontadores de Terrassa.

1.1. Descripción del proyecto

Este proyecto se centra en el estudio de los datos obtenidos de la red de telecontadores de Terrassa y su proceso de ETL (*Extract, Transform & Load*).

En primer lugar, se hará una introducción a los datos, indicando cuál es su estructura, cuáles son los métodos de adquisición y el pretratamiento que se ha realizado sobre los mismos para poder avanzar en su estudio. Con esto también quedará indicado en qué consiste la red de telecontadores de Terrassa y como se distribuye.

A continuación, se indicará el proceso tras la elección de los diferentes métodos de validación de datos, cual es la motivación y como se ha realizado la validación. Con los datos una vez tratados se hará lo propio con los métodos de reconstrucción. Una vez teorizada la validación y reconstrucción se mostrarán los resultados de los mismos cerrando así el primer gran bloque del proyecto.

Tras el tratamiento de los datos y la obtención de resultados aceptados se abre el segundo gran bloque del proyecto, el cual se centra en la implementación de la BBDD (Base de Datos), los modelos ER y Relacionales de la misma y el desarrollo del código necesario. El bloque cerrará con la muestra de los resultados en este aspecto.

Finalmente, el proyecto culminará con las conclusiones y las posibles mejoras del mismo, añadiendo además un conjunto de anexos donde se podrán encontrar todos los scripts y algoritmos diseñados para el desarrollo del mismo.

1.2. Justificación del proyecto

La constante mejora en todos los sectores de nuestra sociedad conlleva la tecnologización de servicios que, hasta hace no mucho, se estaban realizando de forma manual. Este es el caso de la gestión de las mediciones en los contadores de nuestras ciudades.

En los últimos años, el mercado de la telemetría para este tipo de usos ha crecido a pasos agigantados, hecho que se puede apreciar en la aplicación de tecnologías de IoT (*Internet of Things*) para la medición y tratamiento de datos de lectura de agua.

Según el *Smart Water Metering Market* el valor global del mercado para el uso de telemetría en contadores de agua se encontraba en 5,6M € (5,9M USD) a nivel mundial en 2019 con una proyección de hasta 9,1M € (9,6M USD) para 2024 [1].

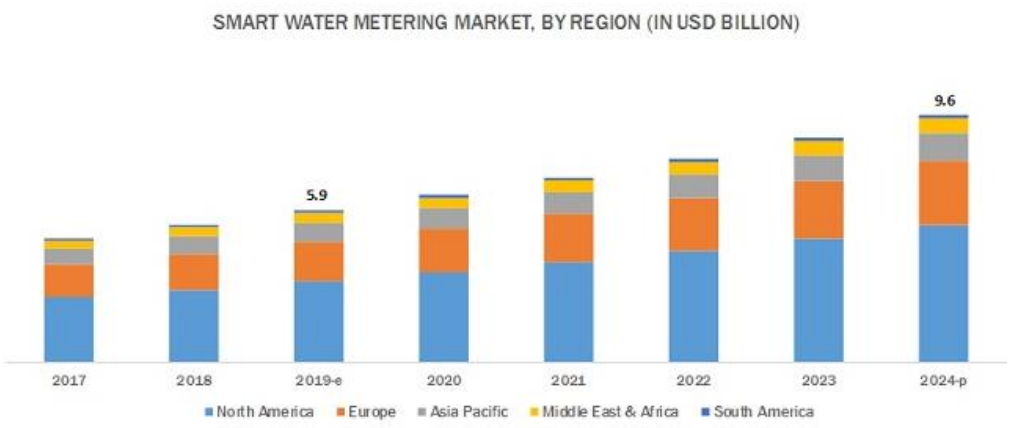


Figura 1.- *Representación del mercado de la telemetría según el Smart Water Metering Market*

Por estos y otros motivos, se aprobó en el pleno de la OAT (Observatorio de Aguas de Terrassa) el 17 de Diciembre de 2021 la moción cuya principal propuesta era:

> "Que el Ayuntamiento de Terrassa fuese consciente de la importancia e impulse la puesta en marcha de un conjunto de propuestas de proyectos estratégicos de innovación del servicio y gestión del agua en Terrassa para que sea el máximo de eficiente, sostenible y optimo, dotando al sistema de un sistema de medida (telemedida) y control de toda la red de agua de sistemas inteligentes, de detección de fugas, de modelos fiables de operación y predicción de nuevas instalaciones que permitan dar un servicio óptimo en calidad y cantidad y, en general, permitan conseguir que el servicio y la gestión del agua municipal sea un ejemplo social y tecnológico a nivel supramunicipal, del país y del mundo." [2]

La moción se presentó en el pleno municipal de Terrassa y se aprobó por unanimidad de todos los miembros del consistorio el 25 de febrero de 2022.

En previsión al incremento que se puede producir en años venideros sobre la telemetría y telegestión en el consumo de agua, es buen momento para buscar nuevos métodos de tratamiento de los datos obtenidos y estudiar la información que nos puede ser proporcionada a partir de estos mismos datos.

Es por este motivo que el presente proyecto se centra en estos métodos de tratamiento y almacenamiento de los datos del sistema y en el estudio de los mismos. Para llevar a cabo este fin, se utilizará el lenguaje de programación Python, dada su versatilidad y coste.

1.3. Objeto del proyecto

El objeto del proyecto es el desarrollo de un conjunto de scripts mediante los cuales se pueda realizar un acondicionamiento de los datos obtenidos de la red (tratamiento, validación y reconstrucción) junto a la creación y alimentación de una estructura de BBDD diseñada.

1.4. Objetivos del proyecto

Los objetivos del proyecto se dividen en dos grupos: el objetivo general y los objetivos específicos:

Objetivo general

- Acondicionar los datos obtenidos de la red de telecontadores de Terrassa mediante la validación y reconstrucción de los mismos y crear una BBDD para la gestión de los mismos.

Objetivos específicos

- Lectura inicial y tratamiento de los datos crudos.
- Validación de los datos crudos aplicando técnicas avanzadas de procesado de datos.
- Reconstrucción de los datos ausentes o detectados como "no válidos".
- Diseño de la base de datos a partir de un modelo ER.
- Implementación de la BBDD mediante sentencias SQL a partir de herramientas de gestión de BBDD en Python.
- Alimentación de la BBDD con los datos procesados en el proyecto.
- Testeo de la BBDD a través de una serie de consultas predefinidas
- Estudio y aplicación de técnicas de *clustering* de datos.
- Automatización del proceso, desde la obtención hasta el almacenamiento.

1.5. Alcance del proyecto

En este proyecto se pretende realizar el desarrollo de algoritmos de programación para el acondicionamiento, validación y reconstrucción de datos utilizando el lenguaje de programación Python.

Por otro lado, se desarrollarán e implementarán scripts para la creación y almacenamiento de los datos ya tratados en una BBDD creada exclusivamente para este proyecto utilizando los lenguajes Python y SQL (*Structured Query Language*).

Se realizarán test y se obtendrán resultados iniciales, intermedios y finales sobre datos reales para cada implementación y se realizarán estudios de los mismos con tal de mejorar cada uno de los procesos realizados.

2. Estado del Arte

El abastecedor principal de agua para la ciudad de Terrassa a lo largo del siglo XX fue la empresa privada MINA que, hasta la década de 2010, se encargaba de la gestión del agua en varias localidades de la zona. A mediados de 2018, el Ayuntamiento de Terrassa culminó el proceso de municipalización del servicio de suministro de agua, pasando a ser operado por la empresa pública TAIGUA (Agua Municipal de Terrassa). [3]

TAIGUA es la entidad pública empresarial dependiente del Ayuntamiento de Terrassa que tiene como objetivo la gestión directa del agua en la ciudad de Terrassa. Después de culminar el proceso de municipalización del agua, la entidad inició sus actividades el 1 de Junio de 2018 y comenzó a prestar el servicio de agua el día 10 de Diciembre del mismo año. Un mes después del inicio de las actividades de TAIGUA, el Pleno municipal de Julio de 2018 aprobó el Reglamento del OAT, el órgano que articula la participación de la ciudadanía en la definición de las políticas y en las decisiones estratégicas que afecten al servicio de abastecimiento. Entre las funciones de la OAT, destaca la presentación de propuestas para el funcionamiento de la entidad pública que gestiona el agua. [4]

La infraestructura de la gestión del agua de Terrassa divide la ciudad en 8 pisos de presión i sectores de distribución. Estos son: Sulleva, Can Colomer, Zona Alta-Grípia, Can Boada, Can Poal, Can Parellada Residencial, Can Parellada Industrial i Can Palet de Vista Alegre.

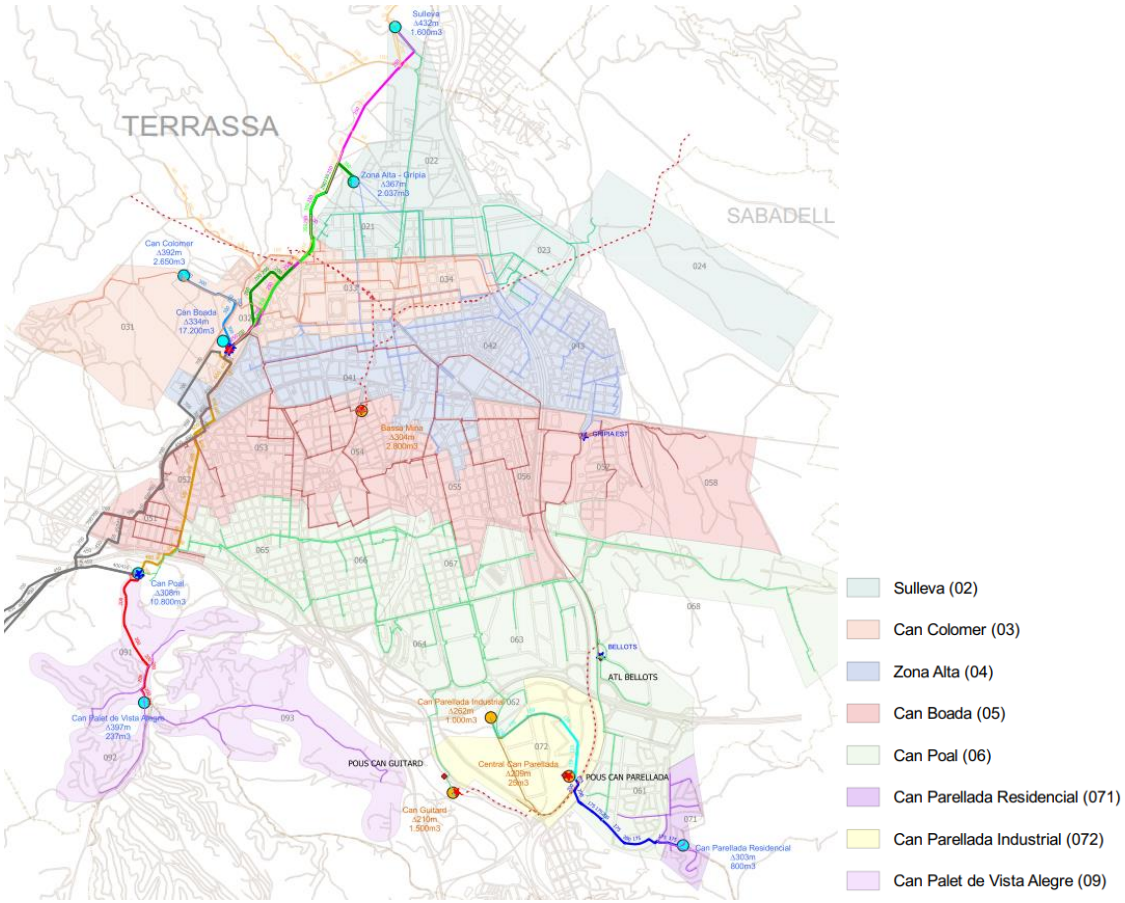


Figura 2.- *Infraestructura de gestión del agua de Terrassa y sus sectores de distribución*

En el año 2021, en Terrassa había un total de 102.084 abonados a la red de TAIGUA, de entre los cuales 99.047 disponían de contadores, la diferencia de número es debido a que los aforos e incendios no tienen contadores.

Actualmente TAIGUA está realizando una prueba piloto de telelectura en la red de suministro de Terrassa para la cual ha instalado 220 telecontadores.

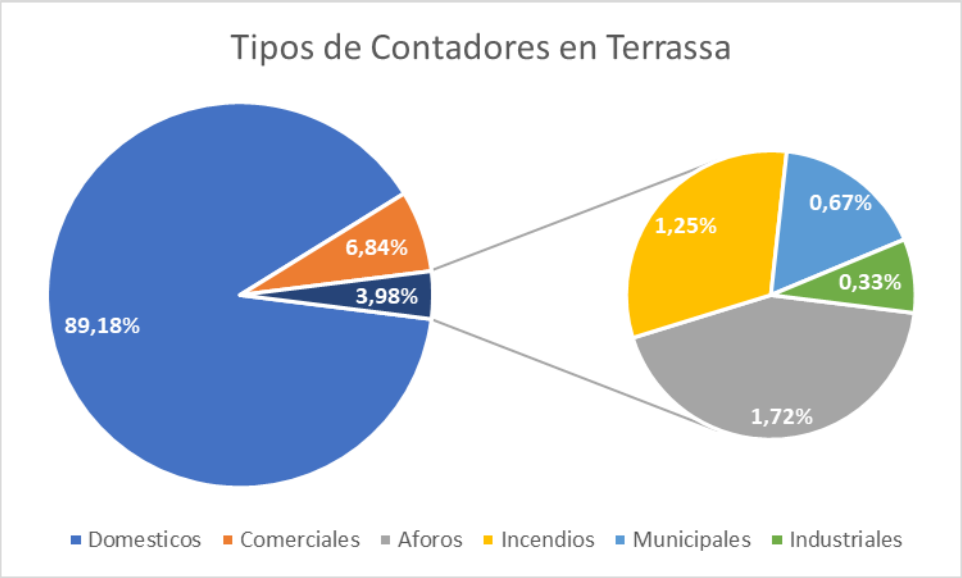


Figura 3.- *Uso de los contadores en Terrassa*

Anteriores a estas adiciones, TAIGUA ya había realizado pruebas piloto previas, disponiendo un total de 467 telecontadores por toda la ciudad de Terrassa. Los datos obtenidos en estas pruebas durante 2019 y 2020 han sido los proporcionados por TAIGUA y sobre los cuales se desarrollará el proyecto.

Estos telecontadores se encuentran situados en los sectores de Zona Alta-Grípia, Can Boada, Can Poal i Can Parellada Industrial (distribuidos entre 12 subsectores), pudiendo ser, en función de su contrato, telecontadores Comerciales, Industriales, Domésticos o Municipales.

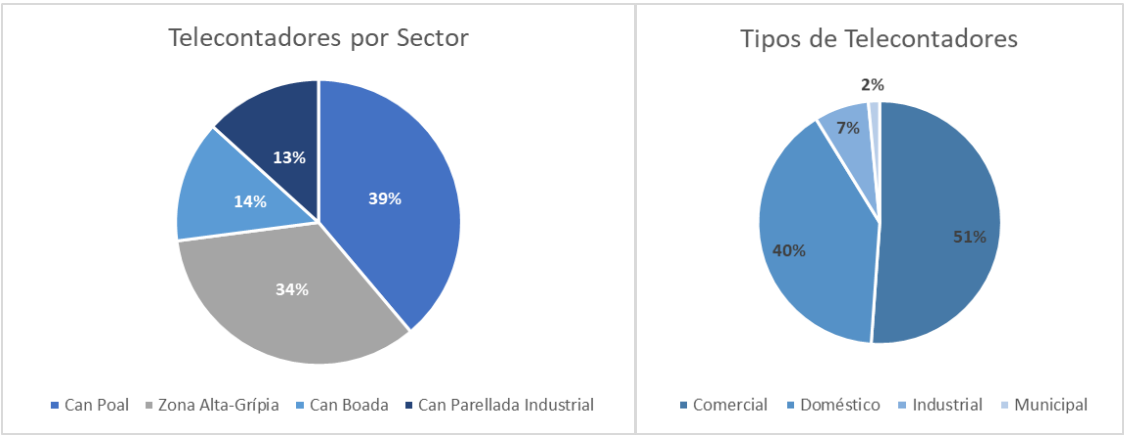


Figura 4.- *(Izda.) Distribución geográfica de los telecontadores en Terrassa.*
(Dcha.) Distribución según el contrato de los telecontadores en Terrassa

Teniendo en cuenta los 467 contadores instalados en anteriores pruebas piloto y los nuevos 220 contadores instalados, obtenemos un total de 687 telecontadores distribuidos por toda Terrassa sobre un total de 102.084 abonados. Por lo tanto, el porcentaje de telecontadores es de tan solo el 0,67%.

Este valor dista mucho del despliegue de telelectura de otros municipios de Catalunya donde, ciudades como Barcelona, Pallejá o Viladecans cuentan con un porcentaje en torno al 50% y el 75% y, otras ciudades como Gavà o Castelldefels del 75% al 100% [5]. Estos datos dan más valor aún a la propuesta del OAT sobre la innovación en el servicio del agua de Terrassa y señalan la necesidad real de la instalación de más telecontadores, los cuales vendrán de la mano de un tratamiento adecuado de los datos de telelectura tanto en validación como en reconstrucción y un almacenamiento de BBDD conforme al sistema.

3. Obtención de datos y acondicionamiento

La estructura de los datos es considerablemente relevante des del punto de vista de la interpretación de los mismos. La forma en la que se distribuyen cada uno de los atributos nos aportan información de los puntos desde los cuales se pueden atacar y como los podemos trabajar.

En este caso contamos con información de cerca de 500 contadores, que registran información horaria durante periodos de 3 meses hasta año y medio. Mediante unos simples cálculos vemos que fácilmente se superan los 3 millones de datos (0,85años x 365días x 24horas x 500contadores = 3.723.000datos para ser exactos). Por lo tanto, se convierte en una tarea de extrema necesidad acomodar los datos a los métodos escogidos para tratarlos de manera eficiente. Este pretratamiento o acondicionamiento de los datos condicionará el trabajo posterior sobre los mismos. Así como también lo hará el lenguaje de programación utilizado.

Python es un lenguaje de alto nivel, dinámico y de código abierto. Estos tres atributos lo convierten en una herramienta de interpretación sencilla y con un gran acceso a librerías y funcionalidades de forma completamente gratuita. Además, la gran velocidad de procesamiento de código, lo hace óptimo para el uso de proyectos relacionados con los macrodatos o *Big Data*, como es el caso.

3.1. Estructura de datos

Existen dos tablas de datos proporcionadas por TAIGUA, en formato CSV (*Comma Separated Values*), a los que se han tenido acceso para el desarrollo del proyecto. Es necesario trabajar con los dos conjuntos simultáneamente para poder interpretarlos de forma correcta.

Como se ha comentado anteriormente, los datos recibidos se obtienen horariamente y quedan registrados. Es este primer conjunto, el registro de datos, el que aportará información sobre las lecturas crudas del sistema. A este tipo de archivo le llamaremos "Archivo de Lecturas", y la información se distribuye de la siguiente forma:

	A	B	C
1	hydro_number	DATETIME	Lectura
2	D03XF063055	01/01/2019 3:41	484772010
3	D03XF063055	01/01/2019 4:41	484772120
4	D03XF063055	01/01/2019 5:41	484772240
5	D03XF063055	01/01/2019 6:41	484772350
6	D03XF063055	01/01/2019 7:41	484772470
7	D03XF063055	01/01/2019 8:41	484772590
8	D03XF063055	01/01/2019 9:41	484772700
9	D03XF063055	01/01/2019 10:41	484772820

Figura 5.- *Ejemplo de datos de lecturas crudas*

- **hydro_number**: Código identificativo del contador. En esta columna se indican todos los contadores a tratar de forma desordenada.

- **DATETIME**: Fecha y hora a la que se han registrado los datos. Los datos en esta columna no siempre son cronológicos y requieren tratamiento.
- **Lectura**: Valor absoluto de la lectura en litros proporcionada por el contador.

Es importante añadir que estos conjuntos de datos no se encuentran en un solo archivo CSV. En este caso se encuentran divididos en dos grupos, el primero de ellos hace referencia a las medidas registradas únicamente en 2019, el segundo es de medidas tanto de 2019 como de 2020.

La segunda tabla de datos indica información sobre el contador, al cual llamaremos "Archivo de Referencia" y los atributos del mismo son los siguientes:

	A	B	C	D	E	F	G	H	I	J	K
1	code	hydro_number	connec_id	hydrometer_category	cat_hydrometer_id	category_type	sector_id	dma_id	code_2	codi_netaq1	codi_netaq2
2	E1A42175	19BA400882	888	Domestic	Comptador individual	CP	67	67	33864	45547	52701
3	E1T76300	19BE045046	1064	Industrial	Comptador individual	CPI	1	1	5251	1460	20948
4	E1186927	07TB004979	1065	Comercial	Comptador individual	CPI	72	72	5239	111396	141
5	E1185443	11TC148600	1067	Industrial	Comptador individual	CPI	72	72	16405	141	4111
6	E1305593	09TA362836	1068	Comercial	Comptador individual	CPI	72	72	16405	141	4111
7	E1311857	14TA173971	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
8	E1317701	15TA043125	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
9	E1322530	16BA285454	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
10	E1312908	14TA155597	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
11	E1312909	14TA135000	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
12	E1312910	14TA134999	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
13	E1313116	14TA155589	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
14	E1313239	14TA135004	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
15	E1316425	15TA015059	10474	Comercial	Comptador individual	ZA-GR	41	41	23405	6189	9055

Figura 6.- Ejemplo de datos de información de los contadores

- **code**: Código identificativo del contrato del abonado.
- **hydro_number**: Código identificativo del contador.
- **connec_id**: Código identificativo de la acometida.
- **hydrometer_category**: Tipo de contrato del contador (Comercial, Industrial, Doméstico o Municipal)
- **cat_hydrometer_id**: Tipo de contador (por defecto "Contador Individual")
- **category_type**: Código identificativo del piso de presión donde está el contrato, contador o acometida.
- **sector_id**: Código identificativo del sector.
- **dma_id**: Código identificativo del sector.
- **code_2**: Códigos internos de la empresa.
- **codi_netaq1**: Códigos internos de la empresa.
- **codi_netaq2**: Códigos internos de la empresa.

Ambas estructuras de datos están enlazadas por el parámetro "**hydro_number**", el cual hace referencia al código identificativo del contador.

3.2. Acondicionamiento de los datos

El acondicionamiento de datos se refiere al proceso de limpieza de datos, normalización de conjuntos de datos y realización de transformación en los datos. Como se ha comentado anteriormente, se trata de un paso crítico dentro del ciclo del análisis y tratamiento de datos y puede implicar actividades como la fusión o unión de conjuntos de datos o, de lo contrario, poner conjuntos de datos en un estado que permita el análisis en fases posteriores [6].

Este acondicionamiento de datos a menudo se ve como un paso de preprocesamiento para el análisis de datos porque puede llegar a implicar muchas operaciones en el conjunto de datos antes de desarrollar modelos para procesar o analizar los datos.

Para esta actividad nos encontramos con dos marcos de datos separados, ambos completamente desordenados (incluso con registros divididos en varios archivos) y ofreciendo información incompleta por separado. El primer paso del preprocesamiento será, por lo tanto, ordenar y distribuir los datos para hacerlos más accesibles y permitir una mejor y más rápida interpretación de cada uno de los elementos que lo componen.

Con tal de trabajar los datos de una forma separada y ordenada se ha decidido iniciar el pretratamiento de estos datos dividiendo los registros de lectura de cada contador por separado, generando un archivo de datos para cada uno de ellos sobre los cuales posteriormente se realizarán los estudios de validación pertinentes. Para poder dividir de manera eficaz la información de lectura por contador se ha utilizado el "Archivo de Referencias" (AR), el cual da información única por cada contador, para generar la lista de todos los contadores de la red. Utilizando esta lista como referencia podemos separar las lecturas del "Archivo de Lecturas" (AL) en función de cada uno de sus contadores en archivos únicos.

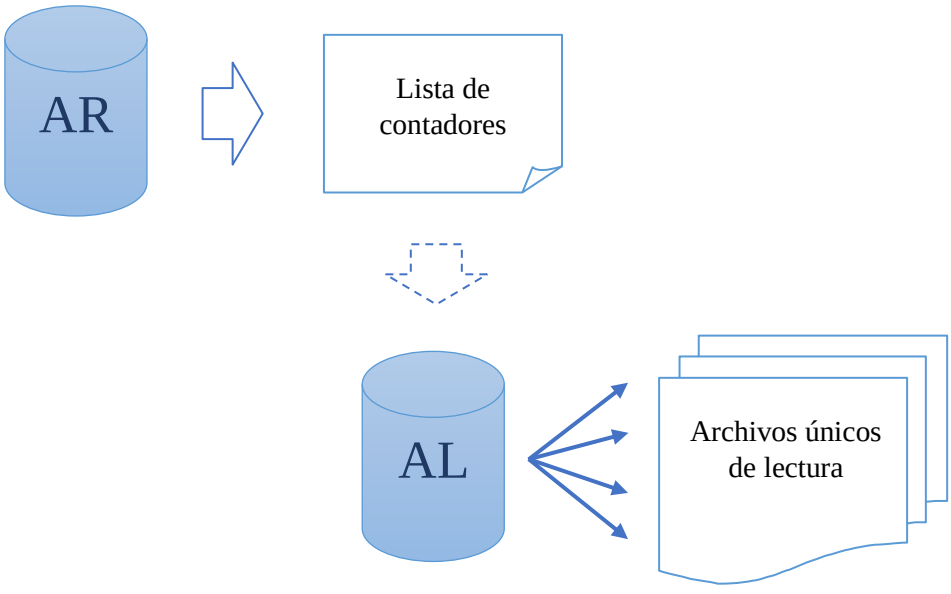


Figura 7.- *Diagrama de pretratamiento de los archivos de datos*

En este punto del proceso del acondicionamiento es cuando se divide cada archivo de lecturas por contador. Al separarlos también se ha convertido el formato de **DATETIME**, que hasta ahora era trabajado como *string* al propio formato de tiempo. También es en este momento cuando se ordenan cada uno de los datos de lectura, obteniendo finalmente una cronología en las lecturas.

Se ha utilizado este método de división de los datos, tomando como referencia el "Archivo de Referencias", ya que se fuerza de este modo a conocer toda la información de cada contador para poder generar un archivo de sus datos de lectura. Utilizar únicamente el "Archivo de Lecturas" para este propósito podría habernos llevado al problema posterior de conocer como se ha comportado el contador, pero desconocer de que tipo es, donde se ubica, etc. Por lo tanto, los contadores que no aparezcan en el "Archivo de Referencias" por el momento se descartarán.

En este punto del acondicionamiento ya nos encontramos con unos archivos con los cuales podemos empezar a trabajar, éstos compartirán una estructura similar con el "Archivo de Lecturas", pero esta vez tenemos por seguro que en cada archivo estamos trabajando con un único contador y que los datos se encuentran ordenados.

	A	B
1	DATETIME	Lectura
2	2019-09-26 06:13:57	3803108
3	2019-09-26 07:13:57	3803118
4	2019-09-26 08:13:57	3803140
5	2019-09-26 09:13:57	3803233
6	2019-09-26 14:13:56	3803682
7	2019-09-26 15:13:56	3803754
8	2019-09-26 16:13:56	3803835
9	2019-09-26 17:13:56	3803901

Figura 8.- *Muestra de datos de lectura absoluta del contador 05TA054679*

Estos nuevos archivos generados constan únicamente de dos columnas, una que indica el momento en el que se ha realizado la lectura y otra en la que vemos la lectura cruda absoluta, que, de aquí en adelante llamaremos "valor integrado".

Estudiando cada uno de estos archivos podemos extraer algunas conclusiones. Con tan solo mirar la primera línea y la última se puede saber, por ejemplo, cual es el marco de tiempo que tenemos para estudiar, con el segundo valor también podemos ver el incremento de valor que ha ocurrido en este periodo de tiempo u obtener el valor integrado inicial y final. Pese a que esta información nos puede ser útil a la hora de buscar patrones o modelos temporales del contador, hay que tener en cuenta el inconveniente que supone no tener valores integrados sincronizados entre varios contadores, ya que, en el caso de querer buscar formas de comportamiento similares siempre estaremos hablando de datos que pese a poder tener una tendencia similar, tienen valores disparejos, ergo no se podrían comparar.

Para solucionar este inconveniente generaremos una tercera columna a la cual llamaremos "valor relativo". En esta columna nos encontraremos con los valores relativos de las lecturas, es decir, trabajaremos con las diferencias entre lecturas y, en el caso de disponer de un archivo en el que no se han perdido datos, serán siempre horarias. Este método de trabajo, aunque puede no

aportarnos información rápida a primera vista, nos será muy útil en varios aspectos que se detallaran en apartados posteriores.

Trabajando con valores relativos podemos encontrar fácilmente caídas en el valor integrado (valores relativos negativos) o incrementos inesperados en el consumo. Por otro lado, nos puede servir de apoyo de cara a establecer modelos de contadores, donde, varios contadores con comportamientos similares, pero con rangos de valores de telelectura muy diferentes coincidirán en su comportamiento relativo.

Al añadir esta tercera columna de datos, obtenemos una estructura como la siguiente:

	A	B	C
1	DATETIME	Integrated	Relative
2	2019-09-26 03:26:47	512268	
3	2019-09-26 04:26:47	512268	0
4	2019-09-26 05:26:47	512268	0
5	2019-09-26 06:26:47	512268	0
6	2019-09-26 11:26:45	512293	25
7	2019-09-26 12:26:45	512293	0
8	2019-09-26 13:26:45	512315	22
9	2019-09-26 14:26:45	512319	4
10	2019-09-26 15:26:43	512321	2
11	2019-09-26 16:26:43	512321	0

Figura 9.- *Muestra de datos de lectura absoluta y relativa del contador 06TA014813*

Con este paso, dispondremos un marco de datos con el que trabajar cómodamente que nos servirá de antesala a procesos de validación y reconstrucción más complejos.

3.3. Visualización de los datos

La representación gráfica de los datos de forma adecuada es igual de importante que su tratamiento. La visualización de datos nos da una idea clara de lo que significa la información al darle un contexto visual a través de gráficos. Esto hace que los datos sean más naturales y, por lo tanto, facilita la identificación de tendencias, patrones y valores atípicos dentro de grandes conjuntos de datos.

Para el caso en cuestión, la representación de los datos ha sido una constante en el desarrollo del proyecto. En varios puntos se han utilizado gráficas con el fin de comprender la situación de una forma mucho más rápida y efectiva y ha servido para atajar problemas a tiempo o para estudiar nuevos métodos de aplicación tanto en la validación como en la reconstrucción de los datos.

La mayoría de las representaciones gráficas de los datos se han realizado utilizando Python, ya que permite la automatización en la obtención de gran cantidad de gráficos, por otro lado, para elementos puntuales se ha utilizado Microsoft Excel como herramienta de acceso rápido a información concreta.

Algunos ejemplos de visualización de los datos acondicionados son los siguientes:

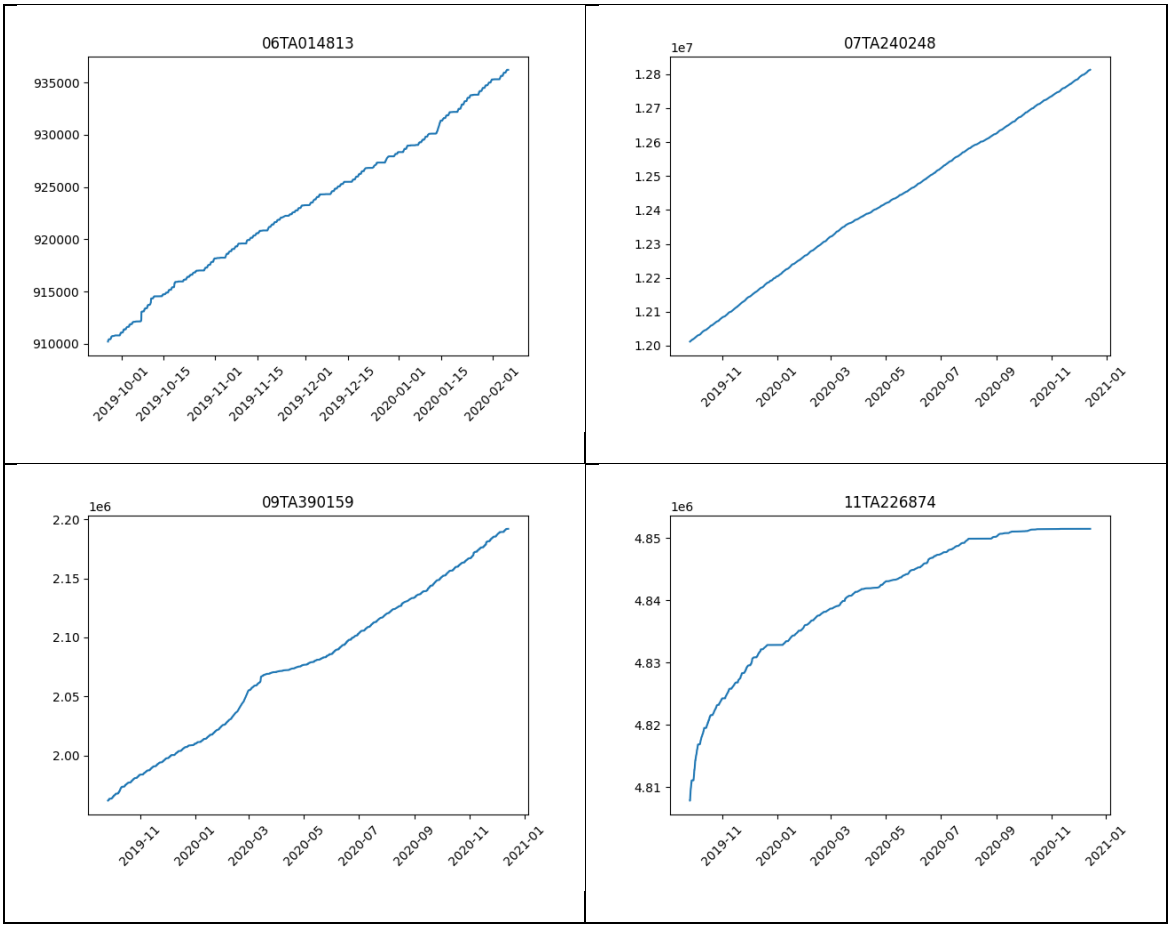
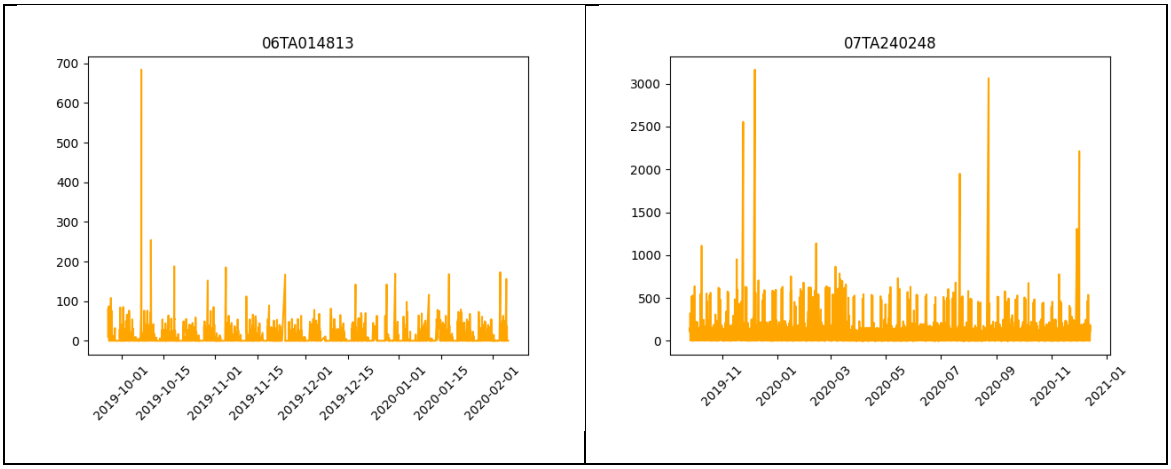


Figura 10.- *Visualización de lecturas absolutas de varios contadores*



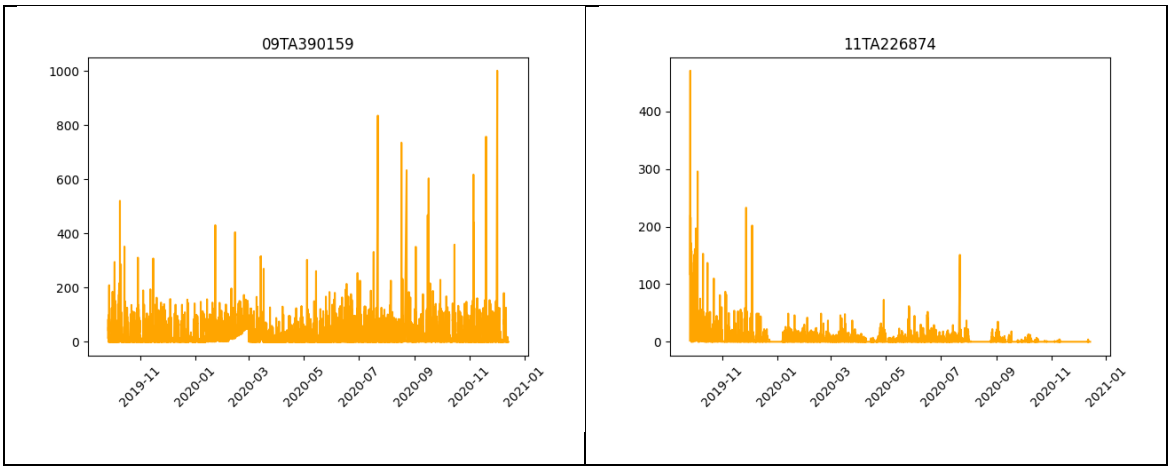


Figura 11.- *Visualización de lecturas relativas de varios contadores*

Una vez han quedado los datos acondicionados y la forma de representación ha quedado definida, se puede iniciar la definición de los métodos de validación empleados sobre los datos.

4. Validación

Se conoce como "validación de datos" al proceso que consigue identificar y distinguir entre los datos fiables de los que no lo son y que precede al proceso de reconstrucción de los mismos. Esto se consigue mediante las conocidas como "reglas de validación", "restricciones de validación" o "rutinas de comprobación".

A través de los mencionados métodos se comprueba la significación, corrección y seguridad de los datos que se utilizarán en el sistema, asegurando de esta manera que los datos sean óptimos para el propósito deseado, además de válidos, seguros y razonables antes de su uso.

4.1. Métodos de validación y gestión de alarmas

El primer paso para poder establecer un conjunto de métodos de validación coherentes para/con el sistema es el estudio exhaustivo de los datos que se van a tratar. Comprender la utilidad de los datos y la información que nos aportan en función de los valores que presentan nos dan las pistas para poder definir cuál puede ser el comportamiento típico, por lo tanto "esperado", para las mediciones de los contadores. Todo lo que se encuentre fuera de estos parámetros puede ser interpretado como un comportamiento atípico, el cual es necesario reportar, estudiar y decidir cuál será el método de intervención con tal de corregirlo.

Los métodos de validación propuestos en esta memoria se han inspirado en la norma AENOR UNE 500540, que define un conjunto de tests para validar los datos de estaciones meteorológicas [7].

Para este propósito se han definido una serie de tests a realizar sobre los datos en función de qué estamos evaluando. Siempre y cuando un conjunto de datos no cumpla con los requisitos establecidos se dispararán ciertas alarmas. Estas alarmas quedarán registradas y, en función de su prioridad se utilizarán (o no) unos u otros métodos de reconstrucción.

Figura 12.- *Diagrama de tests de validación y prioridades*

Si bien es costumbre utilizar métodos de validación relacionados con el formato de los datos, en este caso no son necesarios ya que siempre trabajamos con formatos conocidos. En función de los datos que se tratan para la aplicación se ha establecido el conjunto anterior de tests y el orden indicado en la figura anterior.

Con estos tests se pretende dar nombre a todos los posibles comportamientos anómalos (o al menos a los más relevantes) que podemos encontrar dentro de la estructura de los datos de las lecturas en los contadores.

4.1.1. Dato Perdido

La continuidad en los datos es, probablemente, el condicionante más importante a la hora de poder interpretar las lecturas y encontrar patrones de comportamiento. Por lo tanto, es más importante aún reconocer cuando se están perdiendo datos en el tiempo.

El telecontador consta de dos funcionalidades que definen su comportamiento. Por un lado, realiza la telecomunicación de los datos con el servidor (tele-) y por otro se encarga de llevar el conteo del consumo de agua (-contador). Cuando la función de telecomunicación falla, se pierden datos en el tiempo, pero una vez reestablecida la comunicación con el servidor se recupera el último valor leído y, pese a que si estudiamos los datos temporales de lecturas incrementales podemos no ser capaces de apreciar este fenómeno, si comparamos con las lecturas relativas se puede apreciar un incremento notable, es decir, una diferencia notoria entre dos valores que, teóricamente, debían ser consecutivos.

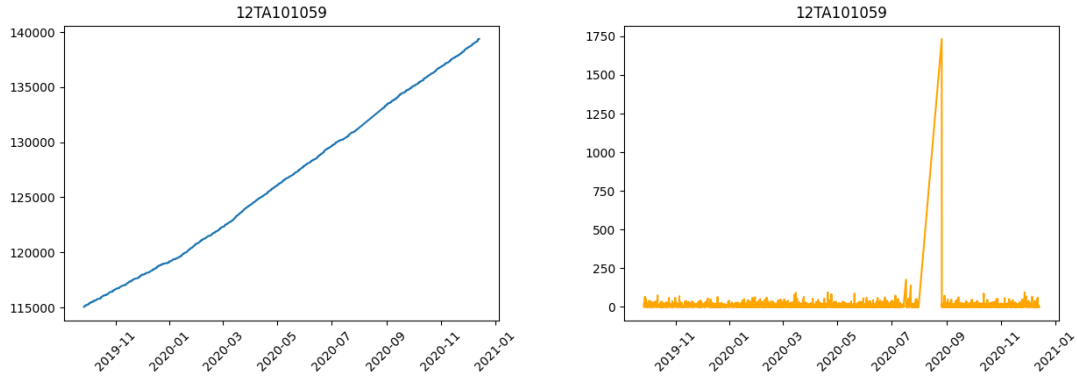


Figura 13.- *(Izda.) Visualización aparentemente correcta de lecturas absolutas. (Dcha.) Visualización de pérdidas de datos en lecturas relativas*

Para determinar si el comportamiento es el adecuado o si se debe hacer saltar la alarma en este test, se debe ir comparando el parámetro "DATETIME" con su valor anterior. Conociendo que los datos deberían estar recibiéndose cada hora, si el tiempo que ha transcurrido entre una medida con la siguiente ha sido mayor a este tiempo, tenemos toda la información necesaria para hacer saltar una alarma.

El espacio de tiempo entre datos es otro parámetro a tener en cuenta. Los procesos de reconstrucción pueden ser diferentes en función de cuánto tiempo haya transcurrido sin recibir ningún dato. Es por este motivo que la alarma de pérdida de datos establece un valor de "gravedad" en función del tiempo transcurrido entre datos (Δt). Los 5 marcos establecidos son los siguientes:

- 1h < Δt ≤ 5h: Nivel de Alarma 1.
- 5h < Δt ≤ 1dia: Nivel de Alarma 2.
- 1dia < Δt ≤ 5dias: Nivel de Alarma 3.
- 5dias < Δt ≤ 15dias: Nivel de Alarma 4.
- 15dias < Δt: Nivel de Alarma 5.

4.1.2. Caída de Valor

Una consideración importante a la hora de entender la estructura de los datos es tener en cuenta que el contador del agua consumida no debe caer en ningún momento. En este caso la lectura es un parámetro integral, es decir, siempre debe aumentar o al menos mantenerse en el tiempo.

Existen dos formas de comprobar que los valores de lectura siempre están en aumento en el tiempo, estas serían; evaluar constantemente todas las medidas y comprarlas con las inmediatamente anteriores para saber si el valor ha caído o, por otro lado, y utilizando las lecturas relativas, en el momento en el que aparezca un valor negativo significará que la medida de lectura ha caído y, por lo tanto, estamos tratando un dato que hay que invalidar.

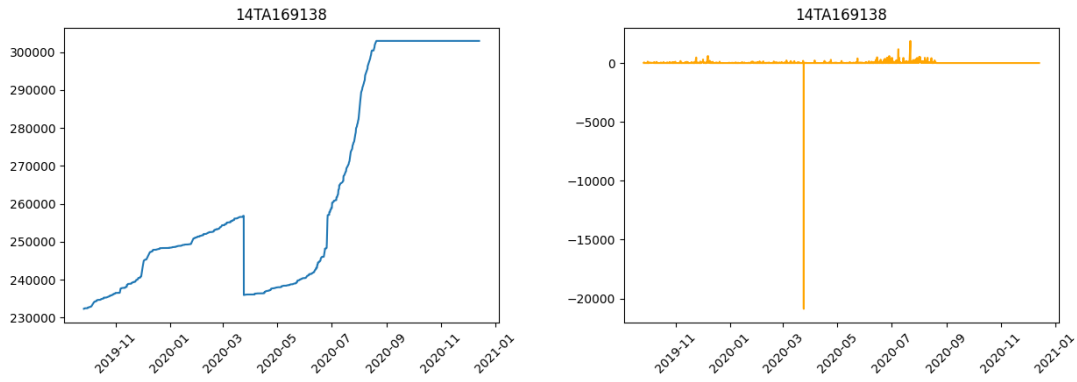


Figura 14.- *(Izda.) Visualización caída en la lectura absoluta.*
(Dcha.) Visualización de valores negativos en la lectura relativa.

Para la implementación de test se ha utilizado el valor relativo de los datos, haciendo saltar un único nivel de alarma en el caso el cual un valor relativo sea menor a cero.

4.1.3. Valor aberrante

En estadística, un valor aberrante o valor atípico es una observación que es numéricamente distante del resto de los datos [8]. Existen varios métodos para hallar estos valores, el recurso más común para ello es obtener el rango intercuartílico (IQR) (diferencia entre el primer cuartil (Q1) y el tercero (Q3)) y sumarlo o restarlo 3 veces al tercer o primer cuartil.

Si la muestra de datos se dividide en cuatro grupos iguales, los cuartiles serían los valores que se encuentran en el punto de división entre secciones, por ejemplo, se puede asumir que el segundo cuartil (Q2) siempre será igual a la mediana de los datos [9] [10].

Ecuación 1.- *Ecuación para cálculo del rango intercuartílico*

$$IQR = Q3 - Q1$$

Los valores aberrantes (q) serian aquellos que cumplan con las siguientes desigualdades:

Ecuación 2.- *Desigualdades de selección de valores aberrantes*

$$q < Q3 - 3 \cdot IQR$$

o

$$q > Q3 + 3 \cdot IQR$$

Para el caso en cuestión este método no es aplicable puesto que los tres cuartiles siempre serán valores muy pequeños. Esto es debido a que la gran mayoría de mediciones relativas serán iguales o muy cercanas a cero.

En cambio, el método utilizado para realizar el test se relaciona con la desviación estándar de las muestras de lecturas relativas. La desviación estándar (σ) es una medida que se utiliza para cuantificar la variación o la dispersión en un conjunto de datos, este parámetro se utiliza, entre otras muchas cosas, para definir la regla empírica, también conocida como la regla 68–95–99.7 [11], sobre la cual se ha inspirado el test de validación.

La regla empírica anuncia:

> "Para una distribución aproximadamente normal, los valores comprendidos en un intervalo de semiancho una desviación típica respecto a la media aritmética de la muestra, abarcan el 68% de los datos; mientras que dos desviaciones de semiancho abarcan el 95%; y tres deviaciones de semiancho incluyen el 99.7%." [12]

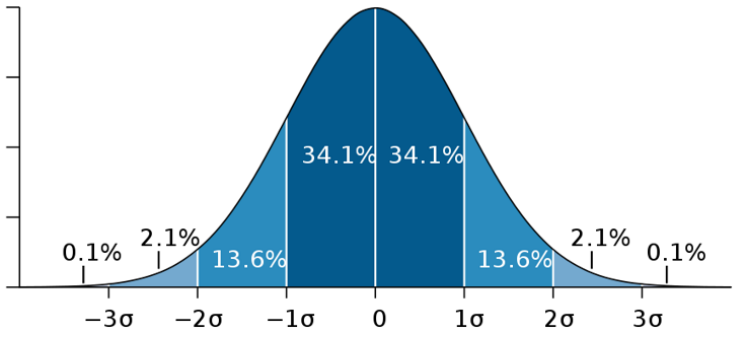


Figura 15.- *Representación gráfica de la distribución normal*

Analizando los histogramas de las lecturas relativas de los contadores podemos apreciar que tienen una estructura similar a la distribución semi-normal (*Half-normal distribution*) [13], que no es más que un caso especial de la distribución normal plegada (*Folded normal distribution*) [14] con la media de valores en cero.

La regla empírica se puede utilizar del mismo modo con una distribución semi-normal que con una distribución normal. Tan solo te debe tener en cuenta que la densidad de cada punto se debe aumentar en un factor de dos al encontrarse plegada justo por su mitad. Por lo tanto, y siguiendo la lógica de la regla empírica obtenemos la siguiente gráfica de proporciones.

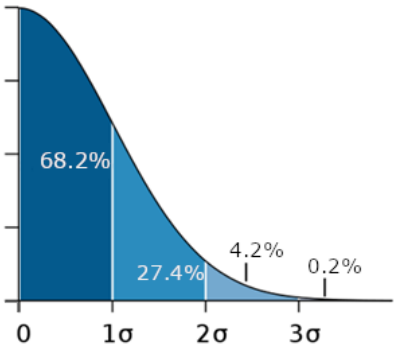


Figura 16.- *Representación gráfica de la distribución semi-normal*

Las siguientes muestras al azar demuestran que la estructura es similar a una distribución semi-normal con una desviación estándar muy baja, esto significa que la gran mayoría de los datos se encuentran cercanos al cero, pero, aun así, existen valores muy alejados de la media que serán los que determinaremos como aberrantes:

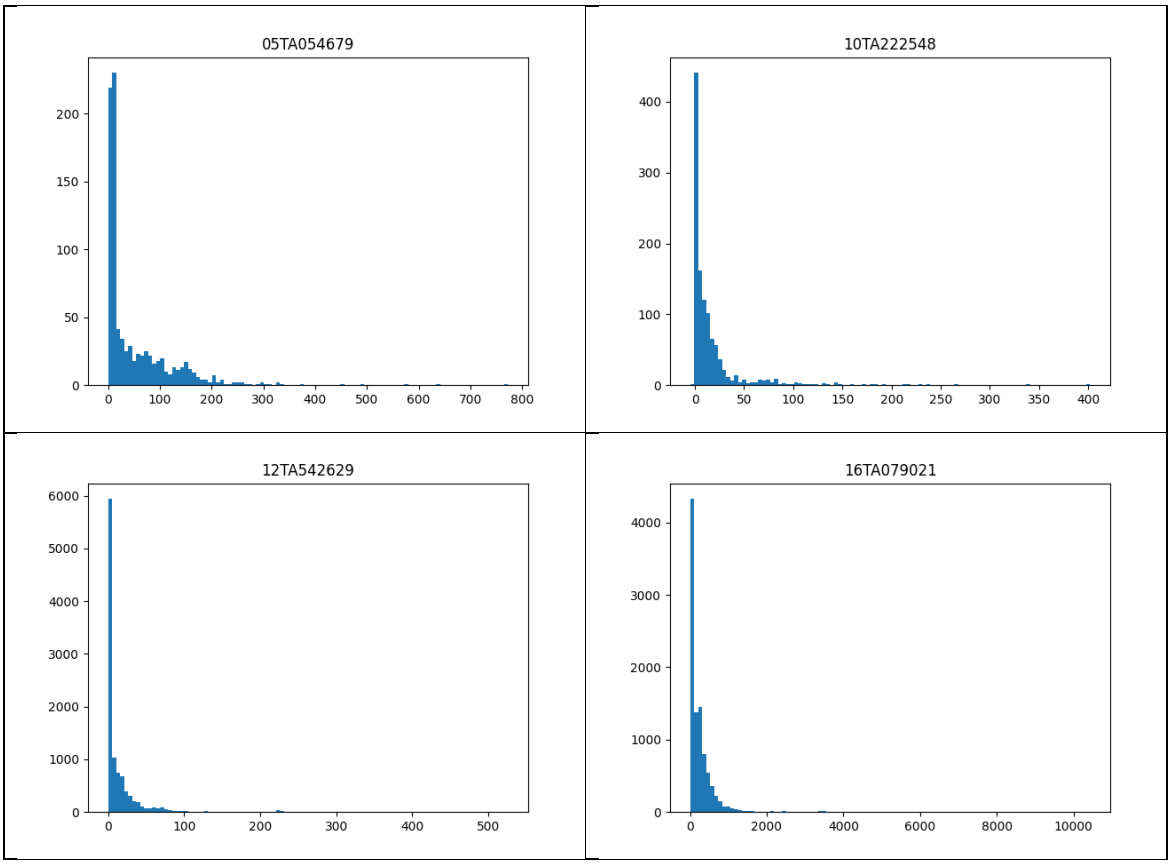


Figura 17.- *Histogramas de valores de lecturas relativas*

Para realizar la criba de valores se ha recurrido a una versión ampliada de la regla empírica y, mediante el análisis de varios factores multiplicativos de la desviación estándar se ha optado por un factor de 6σ.

Tabla 2.- Relación entre los factores de σ y la población esperada

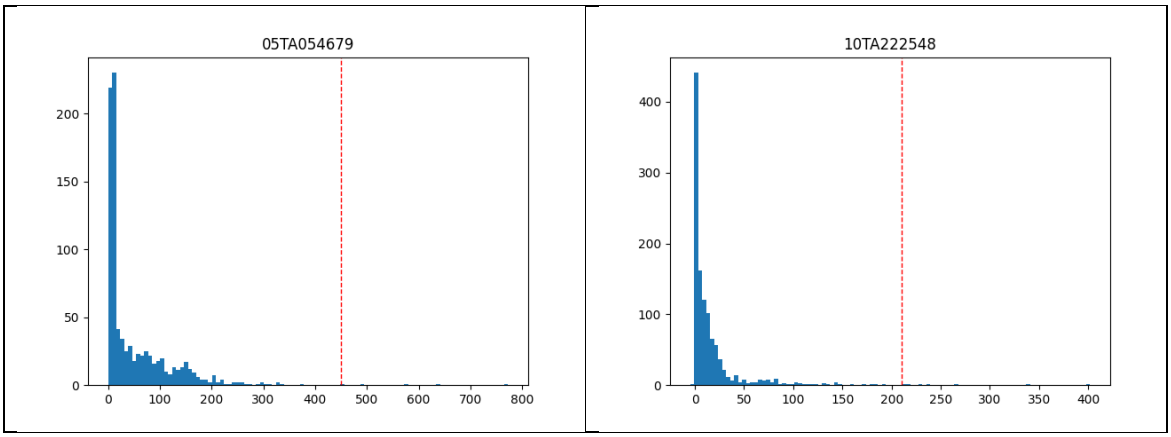
Rango	Población esperada
σ	0.682689492137086
2σ	0.954499736103642
3σ	0.997300203936740
4σ	0.999936657516334
5σ	0.999999426696856
6σ	0.999999998026825

Por lo tanto, la detección de valores aberrantes será una validación dinámica donde, el parámetro que dictamina el valor a partir del cual se hará saltar la alarma variará en función de la muestra que se está estudiando. Para el caso en cuestión se propone no utilizar todos los datos de las lecturas relativas para calcular la desviación estándar. Sino emplear una parte de las muestras para el estudio y, a partir de esta, establecer el umbral de valores aberrantes que posteriormente se aplicaran al conjunto global.

Esta operación es común en todos los modelos de aprendizaje supervisado y ML (*Machine Learning*) y consiste en la división del conjunto de datos en, al menos, dos partes: una parte de entrenamiento (*Train*) y otro de evaluación (*Test*).

Para al cálculo del umbral de validación se ha utilizado un 30% (aleatorio) de los datos. Idealmente se hubiese utilizado una ventana de tiempo delimitada de 6 meses o 1 año, pero vista la estructura de tiempo, no siempre fija, de las lecturas de contadores, no resulta conveniente utilizar un parámetro fijo. Utilizando el 30% de los datos se calcula la desviación estándar de la muestra y, a partir de esta, 6σ.

Aplicando este método sobre las muestras vistas anteriormente, podemos apreciar como el valor umbral varía para cada muestra y se adecua al rango de valores que se tratan:



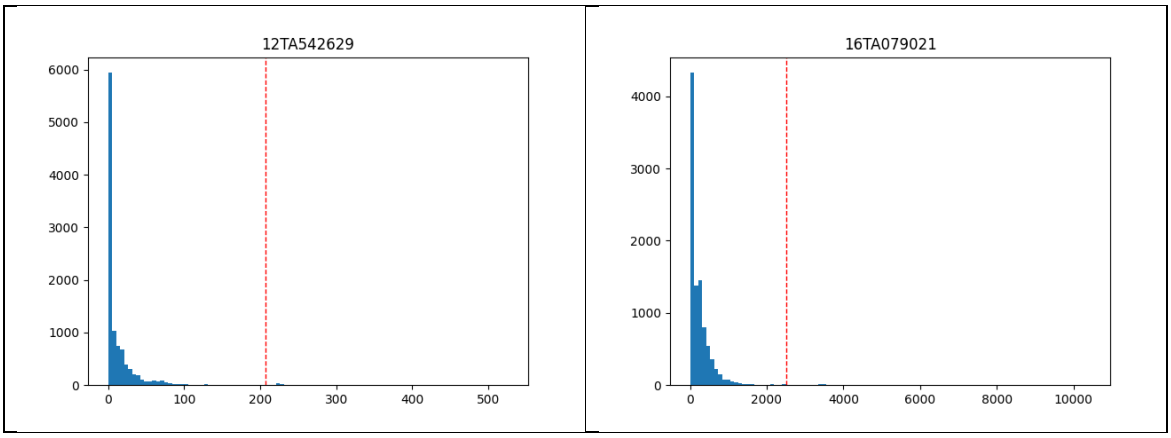


Figura 18.- *Histogramas de valores de lecturas relativas y su correspondiente umbral (6σ)*

Es importante, para el cálculo de la desviación estándar deshacernos primero de todos los valores negativos, es decir, caídas de valor. Esto es debido a que las caídas de valor generan valores relativos aberrantes negativos y, ya que tratamos las muestras como desviaciones semi-normales no nos interesa tener estos valores que, aparte de ser en sí mismos un motivo de alarma, ensuciarían nuestros cálculos.

4.1.4. Valor mantenido

Tanto este test de validación como el siguiente se centran en intentar gestionar las tendencias de comportamiento que no nos interesan.

Anteriormente se han comentado las dos funcionalidades principales de un telecontador y cómo gestionar el problema de las pérdidas de comunicación mediante la alarma de dato perdido. En el caso en el que la información que se pierde es la del contador, la lógica para encontrar el fallo cambia.

El comportamiento de la lectura integral en el caso de la pérdida de conteo se verá reflejada como un valor mantenido en el tiempo que, en el caso de la lectura relativa corresponderá a un cero mantenido en el tiempo ya que, la diferencia entre los valores consecutivos será nula.

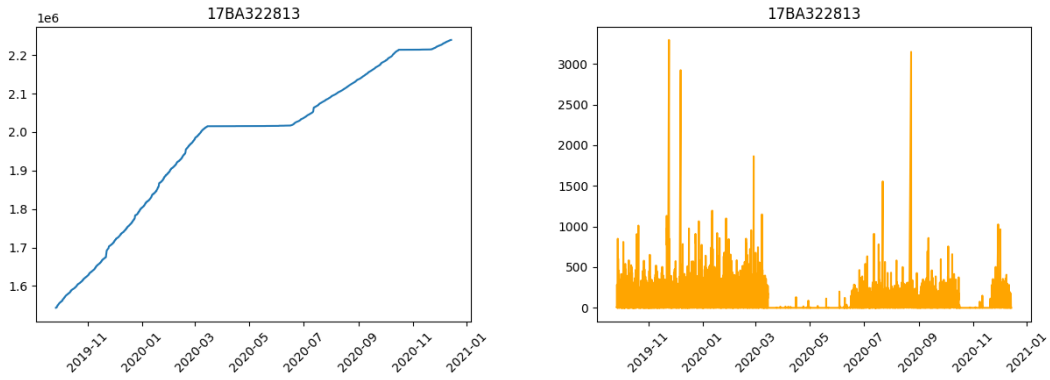


Figura 19.- *(Izda.) Visualización de valor mantenido en la lectura absoluta.*
(Dcha.) Visualización de valor mantenido en la lectura relativa.

El planteamiento de los niveles de alarma para este test concreto dependerá de cuánto tiempo hayamos estado recibiendo el mismo valor, ya que la reconstrucción de los datos puede variar.

En primer lugar, estableceremos una media móvil de diferentes marcos temporales para conocer cuál es el comportamiento de la lectura en el tiempo. En estadística, una MM (*Moving Mean*) es un cálculo utilizado para analizar un conjunto de datos en modo de puntos para crear series de promedios. Así las medias móviles son una lista de números en la cual cada uno es el promedio de un subconjunto de los datos originales. [15]

Por ejemplo, si se tiene un conjunto de 100 datos, el primer valor de la serie de medias móviles podría ser el promedio de los primeros 25 términos, luego el promedio de los términos 2 al 26, el tercer elemento de los términos 3 al 27 y así, hasta por último el promedio de los últimos 25 números del 76 al 100. Para los modelos de datos bajo estudio se ha analizado la media móvil de 5, 15 y 30 días.

Se conoce que la media de consumo de agua diario por habitante es de aproximadamente 120 litros en la ciudad de Terrassa, contando con un hogar promedio de 4 habitantes, obtenemos un consumo horario de 20 litros. Conociendo este parámetro se ha establecido un umbral del 10% sobre la misma, es decir, todo aquel valor que se encuentre por debajo de este parámetro en una media de 5, 15 o 30 días, se tratará como un dato sospechoso que se considerará inválido y requerirá de un estudio posterior y su correspondiente método de reconstrucción. Esta alarma consta de 3 niveles:

- MM (5 días) ≤ 10%20: Nivel de Alarma 1.
- MM (15 días) ≤ 10%20: Nivel de Alarma 2.
- MM (30 días) ≤ 10%20: Nivel de Alarma 3.

4.1.5. Aumento Abrupto

El último de los tests a realizar sobre los datos vuelve a centrarse en la tendencia de los datos, en este caso, se ha estudiado el método de gestionar un aumento abrupto en la lectura de consumo del contador.

En este caso, un aumento abrupto, haría referencia a un incremento prolongado del valor de lectura en un corto periodo de tiempo que no sigue la tendencia del resto de la muestra. Este fenómeno puede no siempre deberse a un mal funcionamiento del sistema, al igual que el valor mantenido también puede tener un motivo, no obstante, son situaciones las cuales se deben tener en cuenta y pueden servir para dar información la muestra y llegar a ser importante para conocer más aun el sistema estudiado.

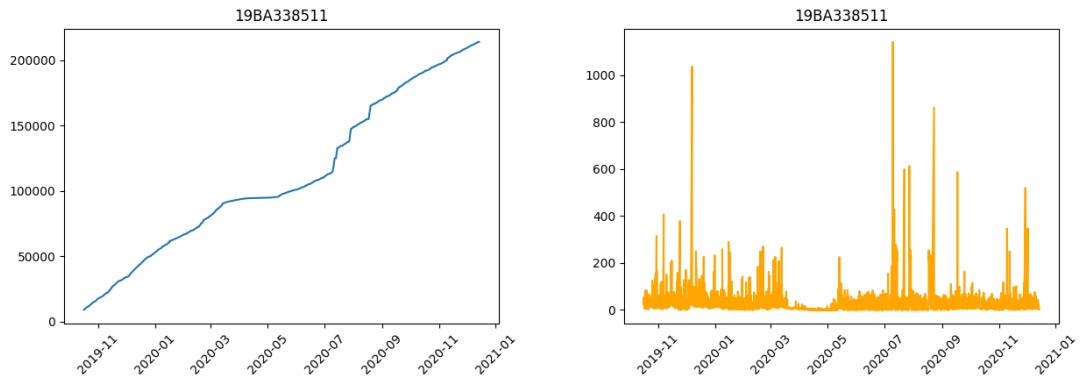


Figura 20.- *(Izda.) Visualización de aumento abrupto en la lectura absoluta. (Dcha.) Visualización de aumento abrupto en la lectura relativa.*

En este caso partimos del mismo planteamiento que en el test de validación anterior. Conociendo cual es la media aproximada de consumo por persona por día en Terrassa podemos establecer cual es un valor fuera de tolerancias.

Utilizando también la *MM* de los datos, se ha preestablecido que todo aquel valor superior a 200% de la media de Terrassa se encontrará por encima del umbral aceptado y, por lo tanto, hará saltar la alarma correspondiente. Esta alarma también consta de diferentes niveles en función del tiempo que haya estado mantenida:

- MM (5 días) ≤ 200%20: Nivel de Alarma 1.
- MM (15 días) ≤ 200%20: Nivel de Alarma 2.
- MM (30 días) ≤ 200%20: Nivel de Alarma 3.

4.2. Representación de Alarmas

La representación gráfica de las alarmas ayuda no tan solo a verificar que el proceso de validación resulte según lo esperado, también aporta información visual del sistema, siendo así más sencillo reconocer los posibles fallos del contador que estamos observando

Esta tarea sirve de ayuda en la ideación y desarrollo de los posibles métodos de reconstrucción que sucederán a la validación, pero no aportan un valor añadido en sí mismas a la lógica del ETL implementado.

Con tal de hacer comprensible la aparición los valores que pueden llegar a ser reconstruibles, se disparan las alarmas en forma de líneas verticales sobre la línea de puntos de los valores obtenidos directamente por el contador o tras el preprocesado de los datos. A continuación, algunos ejemplos:

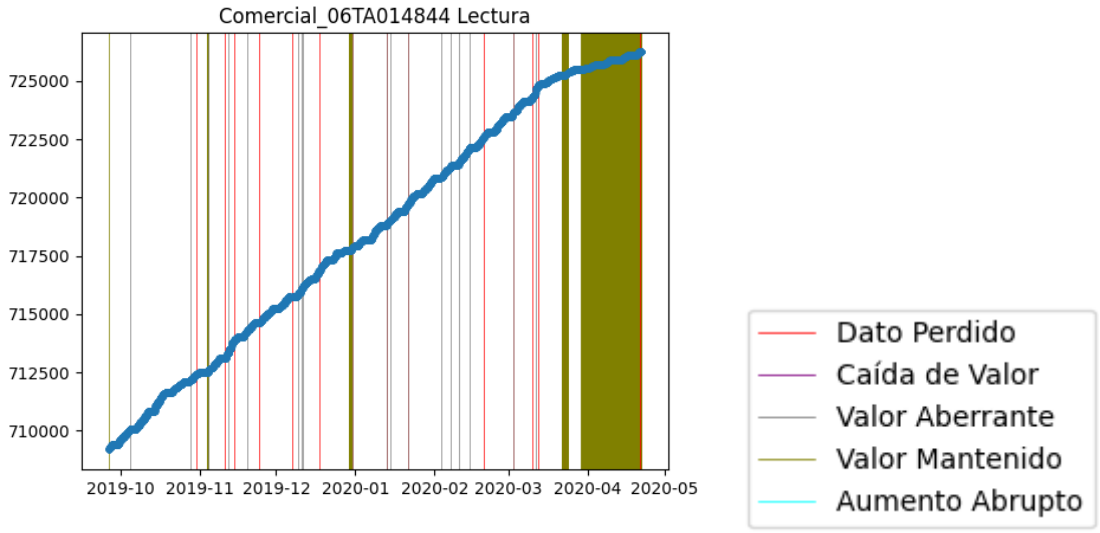


Figura 21.- *Alarmas en las lecturas absolutas del contador 06TA014844*

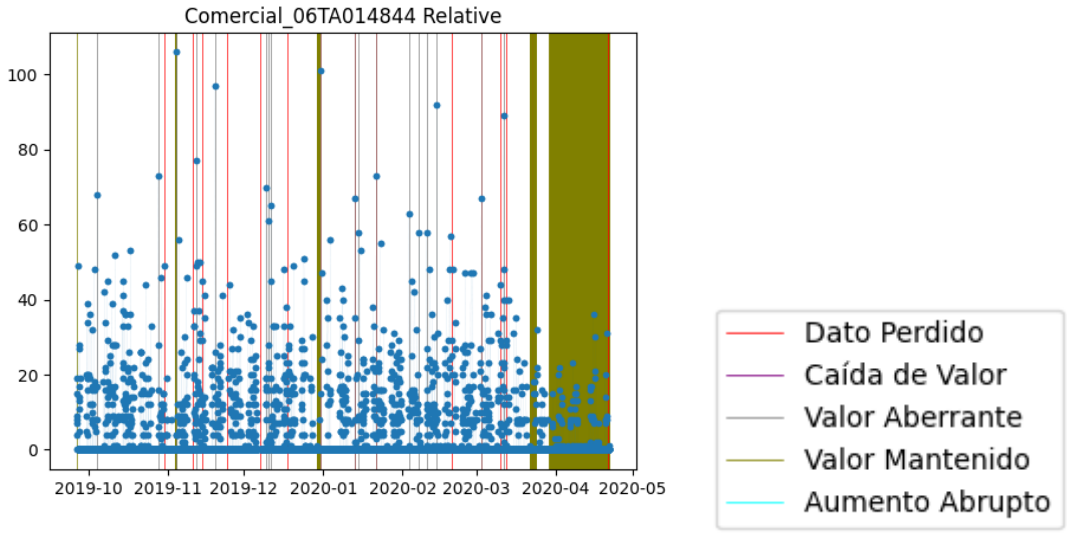


Figura 22.- *Alarmas en las lecturas relativas del contador 06TA014844*

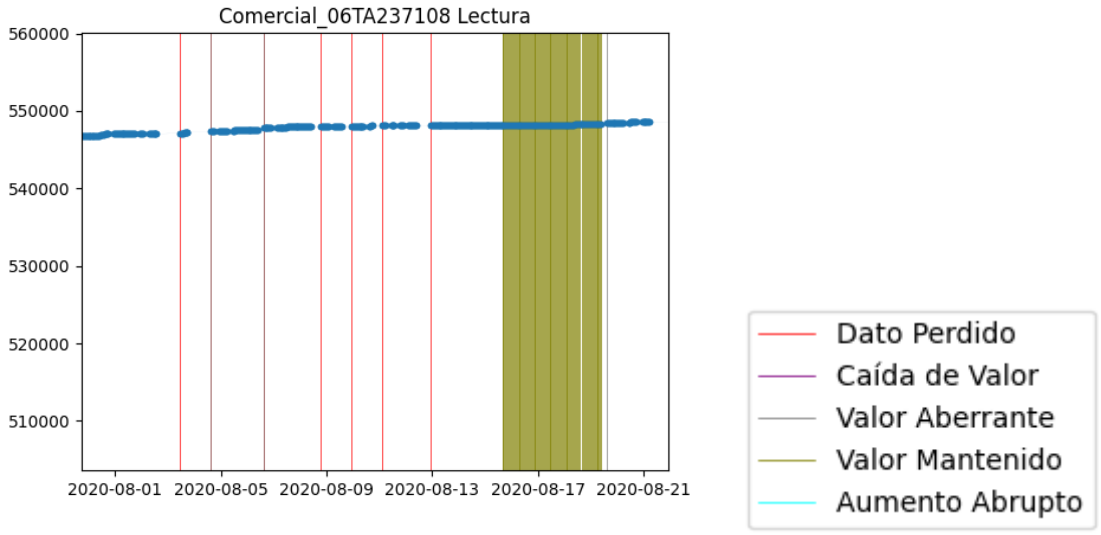


Figura 23.- *Detalle de las alarmas en las lecturas absolutas del contador 06TA237108*

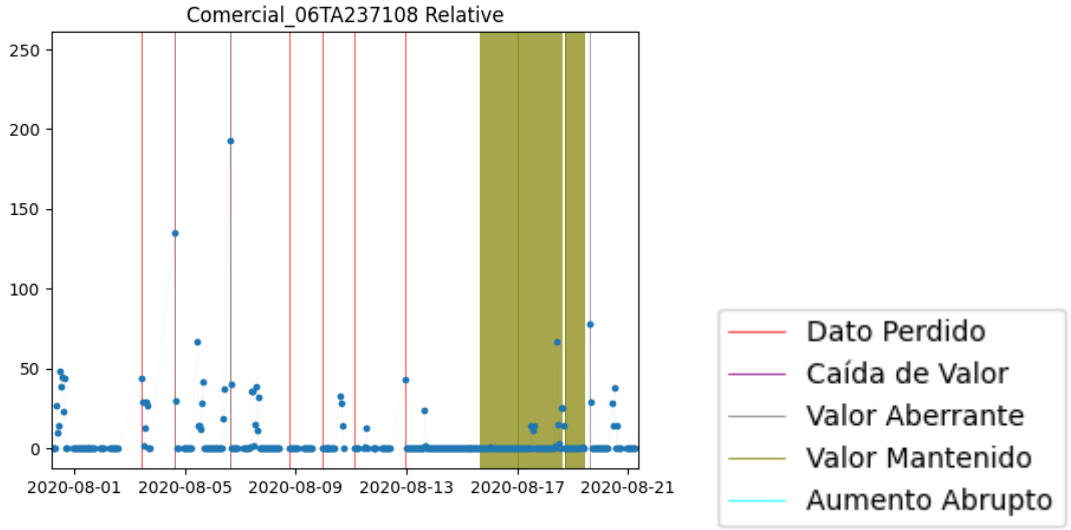


Figura 24.- *Detalle de las alarmas en las lecturas relativas del contador 06TA237108*

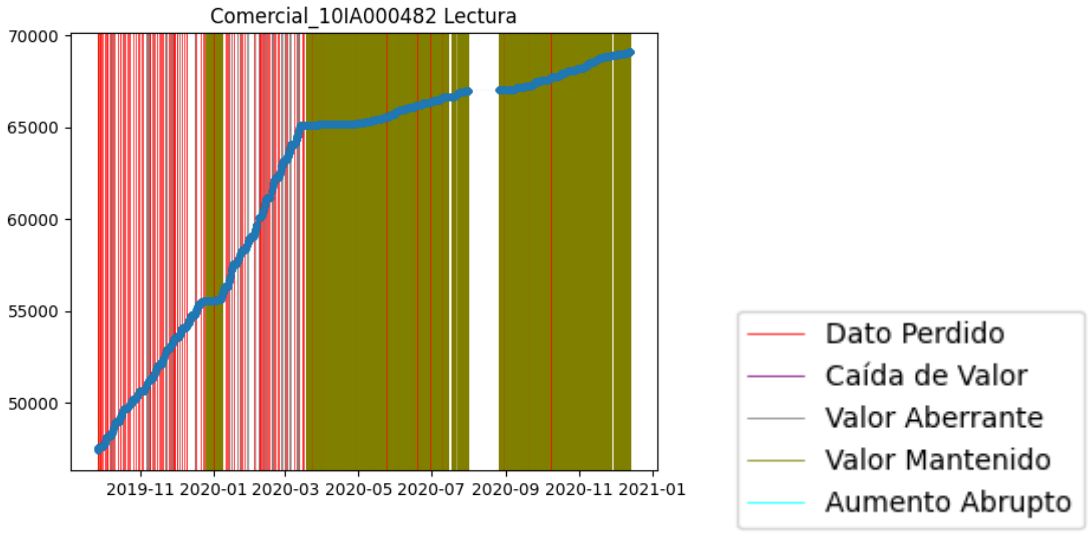


Figura 25.- *Alarmas en las lecturas absolutas del contador 10IA000482*

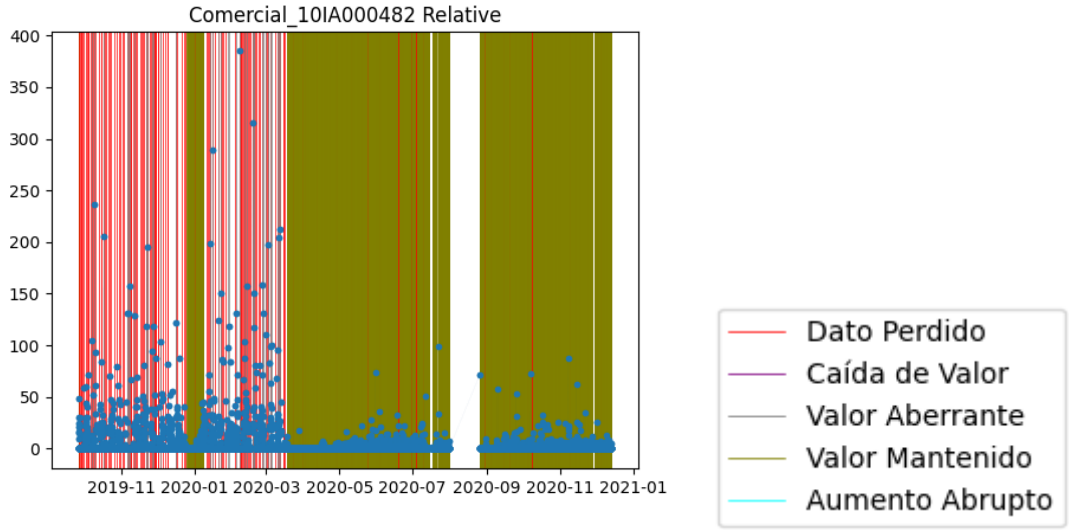


Figura 26.- *Alarmas en las lecturas relativas del contador 10IA000482*

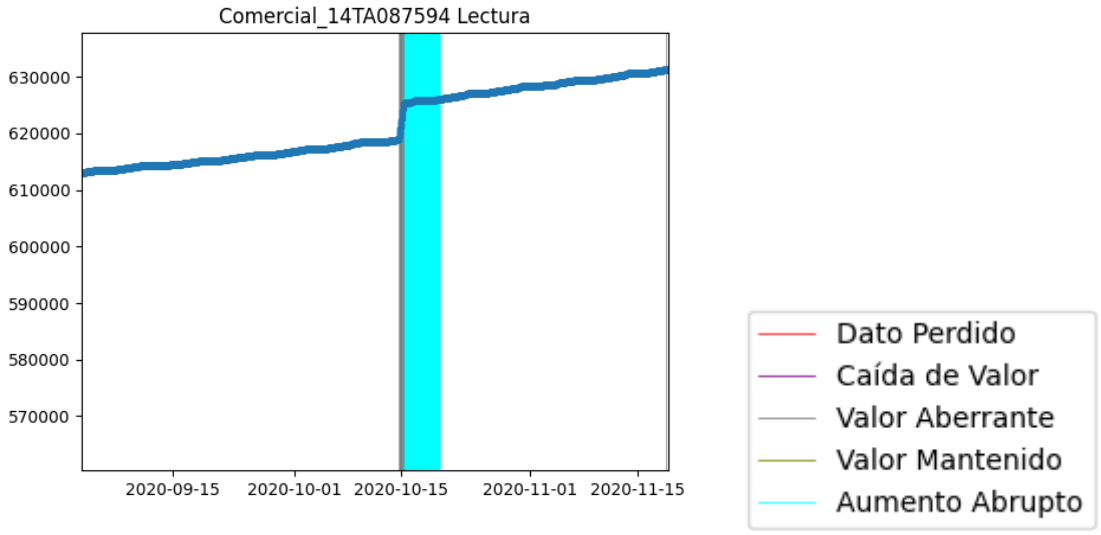


Figura 27.- *Detalle de las alarmas en las lecturas absolutas del contador 14TA087594*

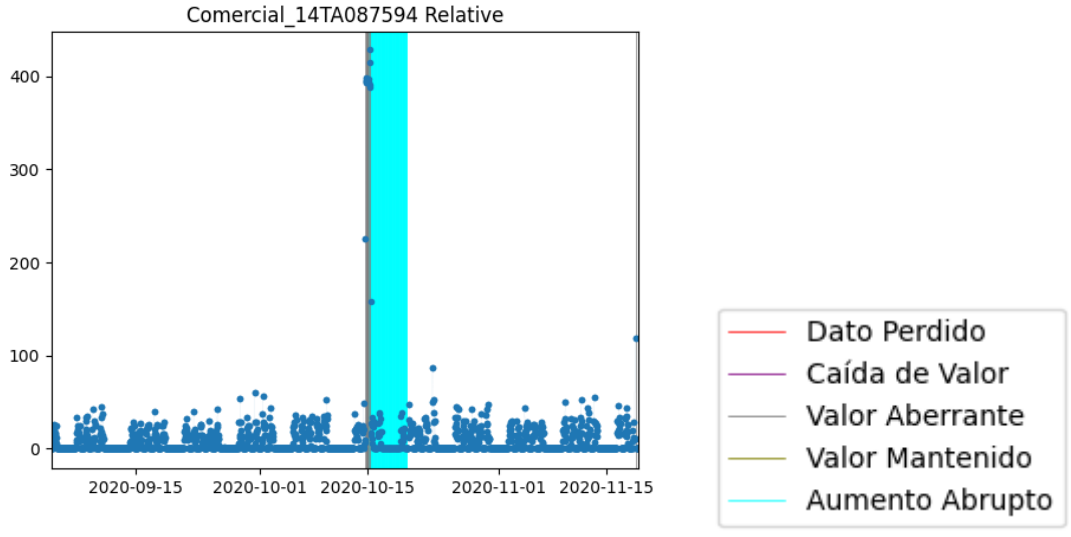


Figura 28.- *Detalle de las alarmas en las lecturas relativas del contador 14TA087594*

5. Reconstrucción

El paso natural que sigue a la validación es la reconstrucción de los datos invalidados. Se entiende la reconstrucción de datos como el proceso consistente en la recuperación de datos que no son accesibles, restaurándolos en un formato accesible.

En nuestro caso, el conjunto de alarmas definidas en el proceso de validación son las encargadas de señalar los datos a tener en cuenta y, en función del criterio escogido, establecer si es necesario reconstruir o, por lo contrario, no merece la pena.

La siguiente tabla hace referencia al resultado del criterio establecidos para reconstruir los datos:

Tabla 3.- Criterios de Reconstrucción

Alarma	Nivel	Reconstruible	No Reconstruible	Método
Dato Perdido	1	X		Interpolación
	2	X		Interpolación
	3	X		Interpolación
	4	X		Modelización
	5	X		Modelización
Caída de Valor	1	X		Imposición
Valor Aberrante	1	X		Interpolación
Valor Mantenido	1		X	
	2		X	
	3	X		Modelización
	4	X		Modelización
Aumento Abrupto	1		X	
	2		X	
	3		X	
	4		X	

De entre todas las alarmas que se han levantado en el proceso de validación, se ha decidido que se reconstruirán los datos relacionados con Datos Perdidos, Caídas de Valor, Valores aberrantes y Valores mantenidos y, en este último caso, solamente las que indiquen un largo periodo de tiempo con el valor mantenido.

La justificación de la no-reconstrucción de estos datos es la misma para ambas alarmas: No se puede afirmar que el fallo se ha producido en el sistema.

Para el caso del valor mantenido, las alarmas saltaban a 5, 15, 30 y 90 días, pero, un valor constante de 5 o 15 días consecutivos puede significar desde una vivienda con sus inquilinos de vacaciones o un establecimiento que ha cerrado por alguna festividad. Por lo tanto, tan solo se han optado por reconstruir todos los datos en el caso en que el periodo el cual el valor se haya mantenido sea igual o superior a 30 días (en el caso de ser así se reconstruiría todo el conjunto de datos mantenidos, no solo a partir de esa cantidad de tiempo).

De la misma manera, el caso del Valor Abrupto puede reflejar una situación factible cómo, por ejemplo, puede ser el caso de la incorporación de un nuevo inquilino en una vivienda (que hará aumentar el consumo con respecto a los datos anteriores). Por estos motivos ambos datos quedan fuera del proceso de reconstrucción y se mantendrá el valor sensado por el contador.

5.1. Métodos de reconstrucción

El orden seguido para la reconstrucción de los datos imita el orden de prioridades a la hora de validar los datos. En todo el conjunto de datos validados existen valores los cuales han hecho saltar varias alarmas simultáneamente, la prioridad en este caso establecerá el método de reconstrucción al cual se verá sometido el dato.

Al no encontrarnos con la misma cantidad de alarmas levantadas y datos a reconstruir, las prioridades finales son las siguientes:

Figura 29.- *Diagrama de prioridades de alarmas en la reconstrucción*

Sobre estos datos, los métodos de reconstrucción han sido tres: Interpolación, Modelización e Imposición.

5.1.1. Interpolación

El método de la interpolación se ha utilizado tanto en el caso de Datos Perdido (para niveles de alarma inferiores a 4, es decir, pérdidas de datos inferiores a un día) como para Valores Aberrantes.

El tipo de interpolación utilizado para estos casos ha sido la interpolación lineal, la cual se utiliza para aproximar un valor en una función para un punto concreto. En este caso particular, se utiliza un polinomio de interpolación de primer grado que se ajusta a partir de los valores de dos puntos.

- Dato Perdido

En el caso de Dato Perdido, se han hallado los dos puntos en el tiempo entre los cuales no se ha recibido ningún valor con sus correspondientes valores relativos de lectura. Al tener ambos datos se espera que el segundo, es decir, al dato obtenido tras la "recuperación" de la comunicación sea mucho mayor que el primero. Será esta lectura concretamente la cual se deberá integrar escalonadamente.

Para esto se ha generado todo el marco de datos perdido y se ha establecido un valor fijo para la lectura relativa que hará converger la lectura absoluta en el valor esperado.

- Valor Aberrante

Este caso es similar al anterior simplificado, ya que no es necesario generar el marco de datos para toda la serie perdida, sino que únicamente se halla el valor previo a la aparición de la alarma, se descarta el aberrante y se interpola con el inmediatamente posterior.

5.1.2. Modelización

Para los casos en los cuales los datos a reconstruir sean de una serie temporal corta, es recomendable utilizar métodos matemáticos simples como la interpolación con tal de establecer los nuevos valores. En el momento en el que las series de datos a reconstruir aumentan, estos métodos se quedan obsoletos y es necesario aplicar otro más complejos.

La modelización es el proceso de hallar un modelo matemático, estos son los modelos que emplean formulismos matemáticos para expresar variables, parámetros y relaciones para estudiar comportamientos de sistemas complejos ante situaciones difíciles de observar en la realidad [16].

Existen varios tipos de modelos matemáticos, que se diferencian según la pretensión a hacer predicciones del tipo cualitativo o si se pretende cuantificar aspectos del sistema que se está modelizando. En este caso el modelo buscado es un modelo cuantitativo o numérico. Estos modelos usan números, fórmulas y algoritmos para representar aspectos del sistema modelizado que permite representar el proceso físico o los cambios cuantitativos del sistema modelado.

Dado el tipo de datos que tenemos, nos interesa prever comportamientos en función de lo sucedido anteriormente. En este punto entran en juego los modelos autorregresivos.

Los AR (Modelos Autorregresivos) se utilizan para pronosticar variables de interés usando una combinación lineal de valores pasados de la variable. El término autorregresión indica que se trata de una regresión variable contra sí misma [17]. Este tipo de modelos son aún más útiles en sistemas que se repiten con periodicidad, como pueden ser lecturas de contadores diarias o anuales.

A nivel matemático, la ecuación polinómica para un modelo autorregresivo de orden *n* se puede expresar de la siguiente manera:

Ecuación 3.- *Ecuación del Modelo Autorregresivo*

$$y_k = c + \phi_1 y_{k-1} + \phi_2 y_{k-2} + \cdots + \phi_n y_{k-n}$$

Dónde y_k es el dato a pronosticar, c es una constante, $\Phi_1, \ldots, \Phi_n$ son los parámetros del modelo y $y_{k-1}, \ldots, y_{k-n}$ son los valores anteriores [17].

Para el caso en cuestión e, igual que sucedió con el umbral de selección del valor aberrante, es complejo establecer un número fijo de parámetros para reconstruir los datos a partir de un modelo de longitud concreta. Tal vez para reconstruir datos perdidos durante 5 días, utilizar 1 día como modelo con, por ejemplo, 24 parámetros (uno por cada hora), puede ser un método acertado de reconstrucción. Sin embargo, si se deben reconstruir 30 días de datos, es posible que el modelo de un único día es insuficiente y puede resultar en modelos no aproximados, necesitando tal vez 15 días para modelar el sistema de forma adecuada.

Del mismo modo generar un modelo de 15 días para tan solo reconstruir 5 se convierte en una tarea ineficiente con redundancia de datos que, para la cuestión, no nos aporta beneficio alguno.

Por lo tanto, para el desarrollo del proyecto se ha optado por establecer un marco de estudio dinámico que variará según la muestra y el punto a estudiar, sobre el cual se modelará el sistema a utilizar para reconstruir los datos necesarios. Se ha establecido que la cantidad de muestras utilizadas sea siempre igual a la cantidad de las muestras a reconstruir y, siempre se tomaran como datos a alimentar los que precedan a la aparición de alarmas.

- Dato Perdido

  En el anterior punto se ha comentado el método de reconstrucción en el caso de pérdida de datos por un periodo de tiempo no superior a 5 días.

  Para este caso, el cambio de paradigma en el periodo de tiempo sin lectura fuerza a un método de reconstrucción más robusto. Es por este motivo que se utiliza AR.

  Con tal de hallar la cantidad de parámetros en la ecuación polinómica del AR, el punto en el cual aparece la alarma y el punto en el que ésta desaparece. Sabiendo que deberá existir un valor de lectura para cada hora se cuentan la cantidad que se encuentran bajo el efecto de alarma. Una vez establecida esta cantidad (*n*), se establece en un tercio el número de coeficientes (*n/3*) y se establece como marcos de datos de entrenamiento (*Train*) la misma cantidad de datos perdidos empezando a contar en sentido inverso desde el último valor de alarma de las lecturas relativas (*n*).

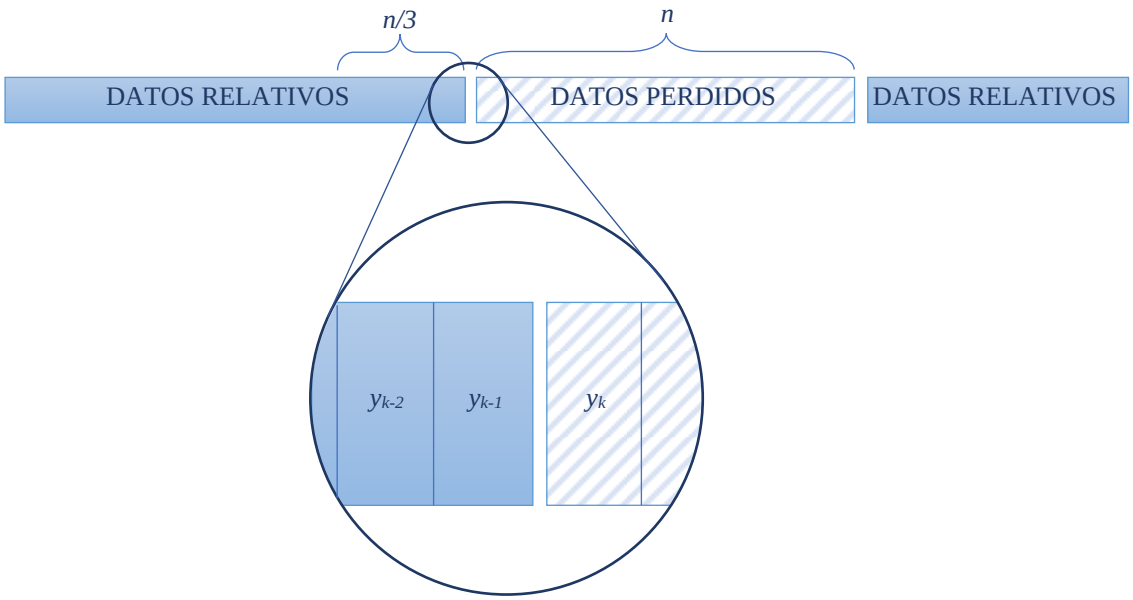


Figura 30.- *Representación gráfica del algoritmo de reconstrucción para datos perdidos*

Gracias a las librerías de autorregresión de Python, se pueden encontrar cuanto valen ***c*** y los parámetros $\boldsymbol{\Phi_1}$, …, $\boldsymbol{\Phi_n}$. Con todos estos datos se puede dar por iniciado el AR para hallar los nuevos valores relativos reconstruidos.

- Valor Mantenido

El algoritmo de búsqueda de parámetros para este caso sigue la misma lógica que para el anterior. La diferencia entre ambas situaciones se encuentra en que, en lugar de hallar los instantes de tiempo en los cuales saltan y desaparecen las alarmas, se evalúa cada momento para encontrar la situación en la cual la alarma es igual o superior a 3 (más de 30 días con un valor mantenido) y cae por debajo de este nivel.

Todo el rango será reconstruido utilizando modelos autorregresivos siempre y cuando el nivel de la alarma haya estado aumentando hasta llegar a un valor de 3 o superior.

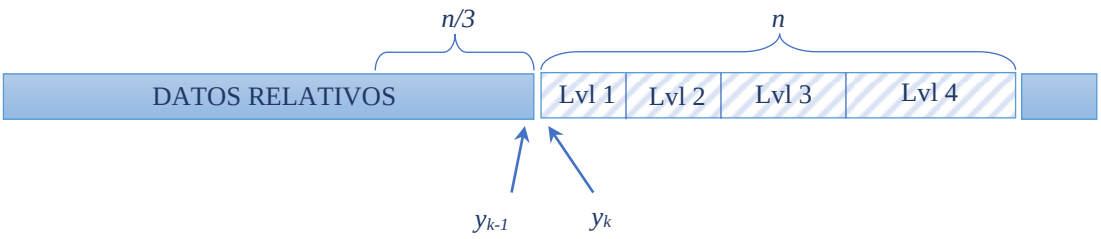


Figura 31.- *Representación gráfica del algoritmo de reconstrucción para valores mantenidos*

5.1.3. Imposición

En casos más singulares puede tratarse de una buena idea el hecho de imponer un valor por la fuerza. Esta práctica, también conocida como *Hard-coding* cuando se utiliza en programación, consiste en incrustar datos directamente en el código del programa en lugar de utilizar los datos obtenidos de una fuente externa (la lectura del contador) o de la reconstrucción mediante modelos matemáticos [18].

La imposición de datos se ha utilizado en el proceso de reconstrucción para los casos de alarmas de caídas de valor. Cómo ya se ha comentado anteriormente, una caída de valor en la lectura absoluta de un contador se traduce en un valor relativo de lectura negativo. Los casos de lecturas relativas negativas son casos aleatorios que se encuentran en pocas de las muestras que se han trabajado, no siguen un orden establecido ni se puede establecer un patrón para detectarlas. Del mismo modo, al desconocer la lectura en un instante aleatorio de tiempo, y con tal de simplificar la actuación sobre las mismas visto su grado de aparición, se ha decidido imponer un valor de cero y, por lo tanto, esa lectura no afectará en los datos posteriores.

5.2. Visualización de la Reconstrucción

El ejercicio de reconstrucción de datos ha sido un proceso dinámico, el cual ha utilizado valores previamente reconstruidos para alimentar modelos que se encargan de seguir encontrando nuevos datos válidos para añadir o sustituir los ya existentes.

Los datos escogidos para la reconstrucción han sido las lecturas relativas debido a que han presentado resultados más favorecedores que la reconstrucción sobre las lecturas absolutas obtenidas directamente del contador. Con las lecturas relativas reconstruidas, el proceso para obtener las lecturas absolutas tan solo necesita del valor inicial de cada uno de los contadores e incrementar su valor en función del valor relativo reconstruido.

A continuación, se mostrarán algunos ejemplos de reconstrucción sobre los cuales ha sido predominante un tipo de alarma por encima del resto (a simple vista) para poder observar el comportamiento de la reconstrucción en cada caso:

- Dato Perdido

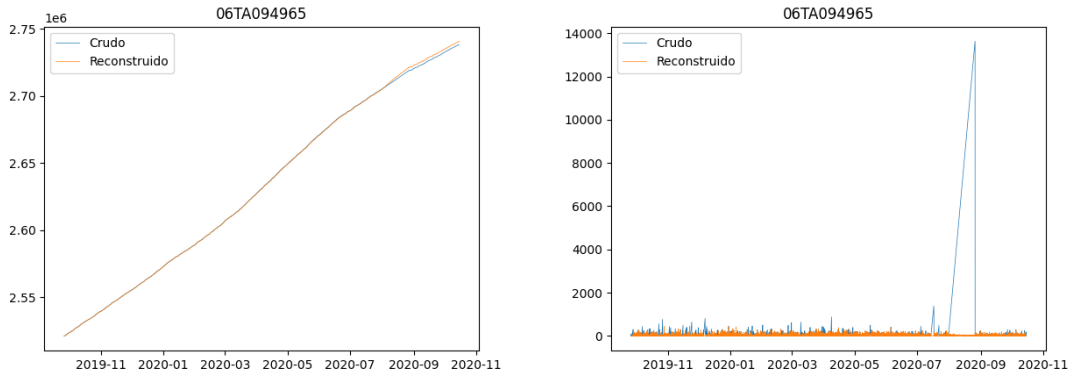


Figura 32.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Dato Perdido) (Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Dato Perdido)*

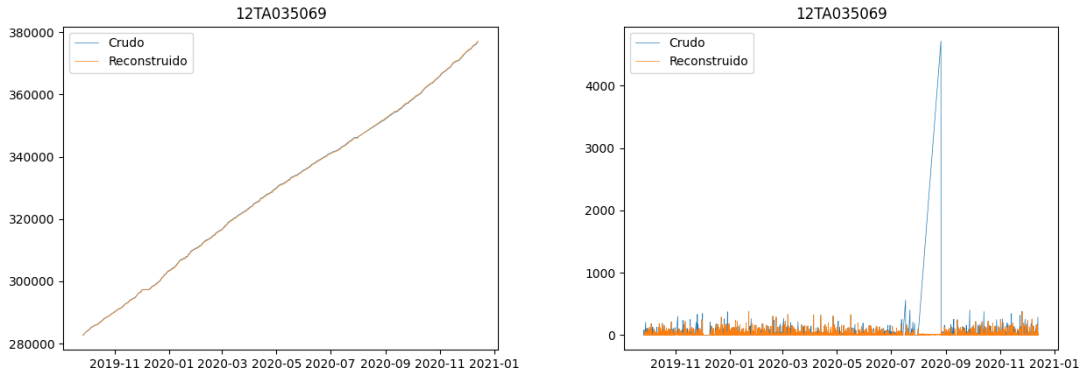


Figura 33.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Dato Perdido)*
(Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Dato Perdido)

En estos casos, la reconstrucción actúa en el momento en el que se detecta que hay instantes temporales (de mayor o menor longitud) en los cuales no se ha obtenido ningún dato. En los casos mostrados podemos ver como el periodo de tiempo el cual el sistema se mantiene sin recibir datos llega a ser de más de un mes, por lo tanto, ha entrado en juego la modelización mediante AR.

Como se puede apreciar, el AR se encarga de continuar con la serie de datos basándose en el periodo anterior, es por eso que en el primero de los casos podemos ver como el valor aumenta ligeramente por encima del valor tomado de nuevo por el contador. Se ha decidido mantener estos valores de reconstrucción por encima de los recogidos por el contador con tal de facilitar la programación ya que en otros casos (como por ejemplo en la segunda pareja de imágenes) la reconstrucción llega a valores prácticamente idénticos a los recogidos por el contador una vez reestablecida la comunicación.

- Caída de Valor

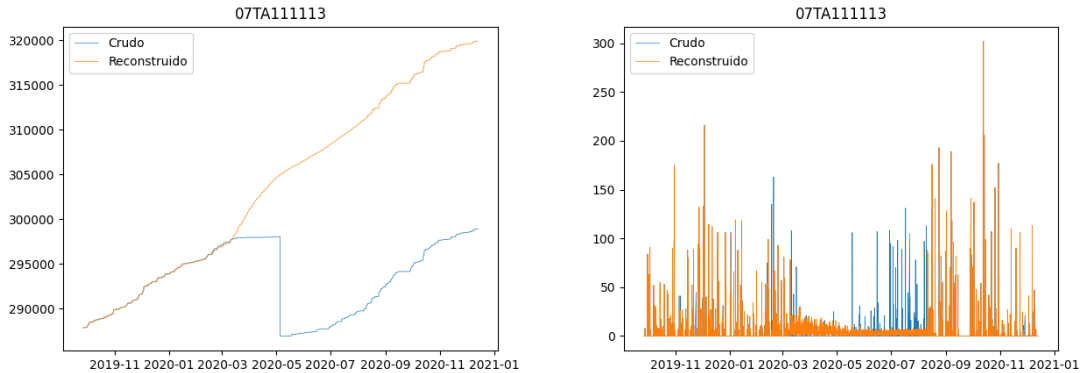


Figura 34.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Caída de Valor)*
(Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Caída de Valor)

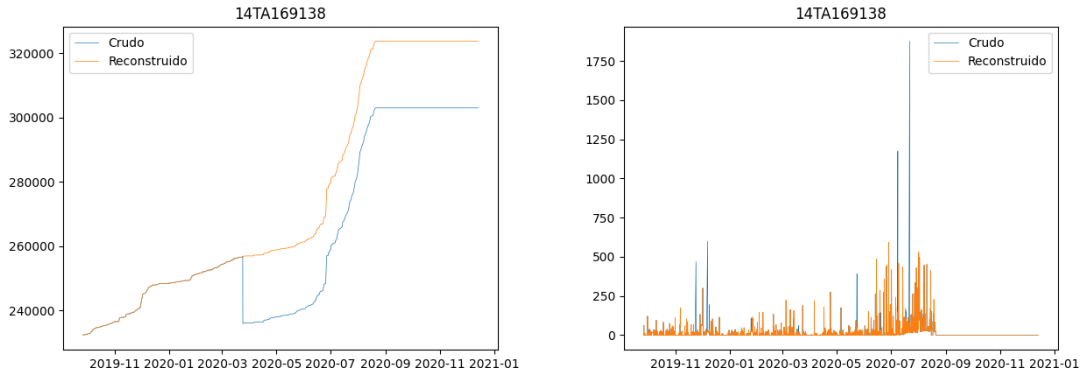


Figura 35.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Caída de Valor)*
(Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Caída de Valor)

En el primero de los casos entran simultáneamente la reconstrucción por valor mantenido y la reconstrucción por caída. Si nos centramos únicamente en la segunda, en ambas parejas de figuras se puede apreciar como la serie continúa como debería ignorando la caída en el nivel.

- Valor Aberrante

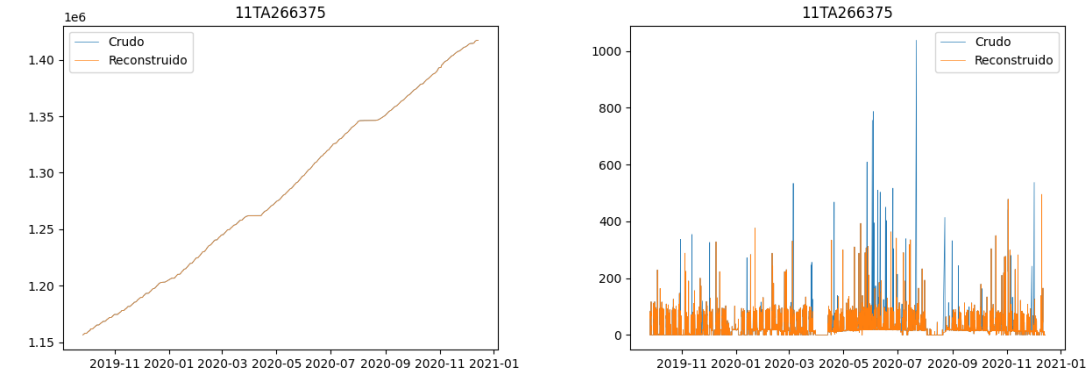


Figura 36.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Valor Aberrante)*
(Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Valor Aberrante)

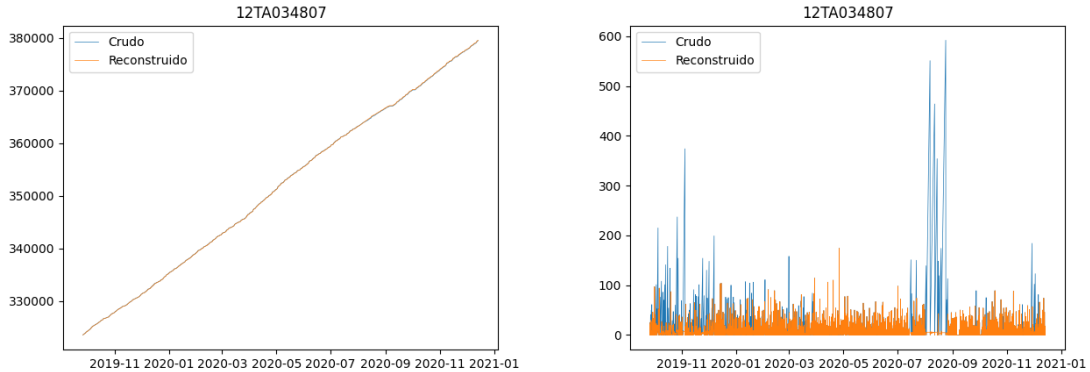


Figura 37.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Valor Aberrante)*
(Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Valor Aberrante)

Mediante interpolación lineal hemos podido deshacernos de los valores extremadamente altos en las lecturas. Para estos ejemplos podemos ver como los picos máximos en los valores reconstruidos llegan a caer por debajo de la mitad de los picos en lecturas crudas mientras que las formas en los consumos absolutos se mantienen completamente igual.

- Valor Mantenido

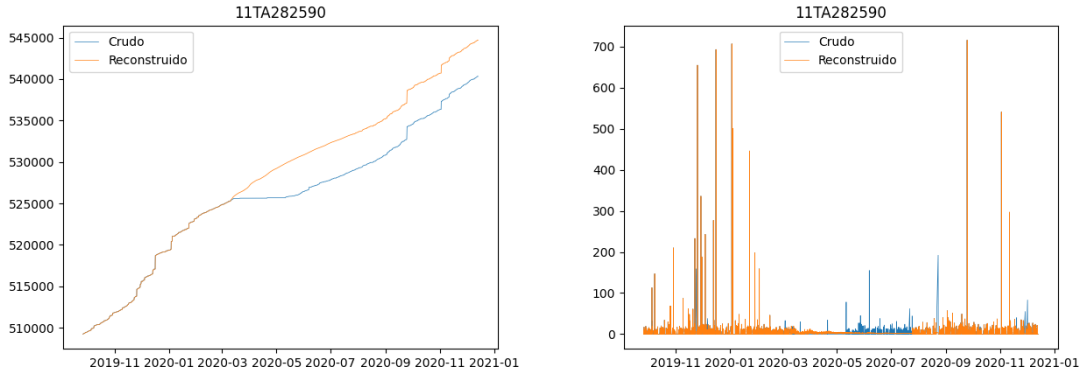


Figura 38.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Valor Mantenido) (Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Valor Mantenido)*

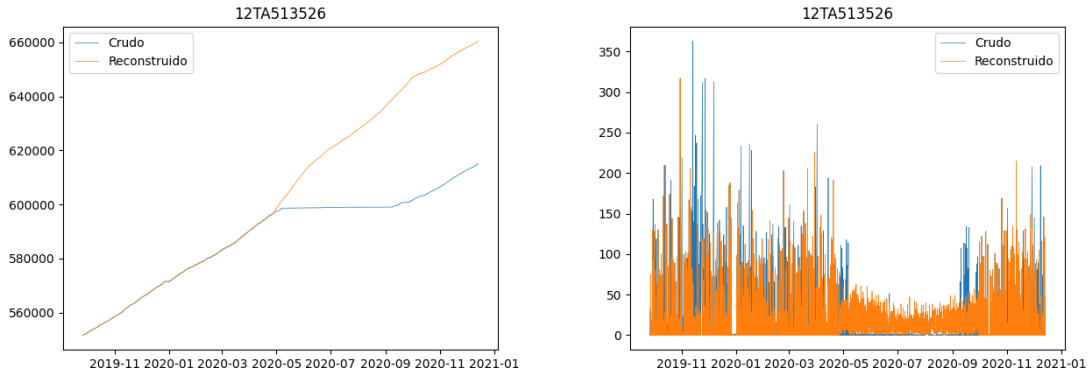


Figura 39.- *(Izda.) Gráfica de datos crudos vs reconstruidos en la lectura absoluta (Valor Mantenido) (Dcha.) Gráfica de datos crudos vs reconstruidos en la lectura relativa (Valor Mantenido)*

En estos casos, al aparecer la alarma de valor mantenido (por encima de 30 días), la reconstrucción de produce desde el mismo instante en el que el valor comienza a estancarse. De nuevo aparece la reconstrucción modelizada mediante AR, la cual mantiene la tendencia en los consumos, sobre todo podemos apreciar esta tendencia en los valores absolutos.

6. Almacenamiento

La información que se ha estado recopilando, validando y reconstruyendo es, en sí misma, volátil. Hasta el momento todo el trabajo se ha planteado para ser desarrollado en un único PC, los datos tratados tienen formato CSV o en su defecto XLSX gestionados en un equipo local, imposible de ser consultados por terceros de una forma sencilla que no implique al usuario inicial generar algún método de almacenamiento en algún servidor.

Las propias siglas del proceso ETL indican que el último paso a tener en cuenta tras el tratamiento de los datos (validación + reconstrucción) deberá ser la carga en el sistema de destino, en este caso una BBDD [19].

Figura 40.- *Diagrama ETL*

Una BBDD se encarga no solo de almacenar datos, sino también de conectarlos entre sí en una unidad lógica. En términos generales, una base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto y, en cuanto a su función, se utiliza para administrar de forma electrónica grandes cantidades de información.

En concreto el tipo de BBDD desarrollada es relacional, estas son las BBDD que cumplen con el modelo relacional, el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información [20].

El primer paso para poder implementar la BBDD es desarrollar el modelo ER (Entidad-Relación) para, a partir de este, hallar el modelo relacional, y finalmente traducirlo al lenguaje SQL. Virtualmente, todos los sistemas de bases de datos relacionales utilizan SQL para consultar y mantener la base de datos [21].

6.1. Modelo ER y Relacional

A nivel teórico existen varios pasos a seguir a la hora de desarrollar una BBDD. La primera de las fases es la recopilación de la información que, en nuestro caso podemos hallar en los "Archivos de Referencia" CSV que se comentaron en el proceso de pretratamiento de los datos.

Una vez obtenida toda la información necesaria, la segunda fase es el diseño del modelo ER, que consiste en plasmar en un diagrama las entidades, atributos y relaciones definidos en los requerimientos recogidos en la fase anterior. Los elementos básicos del modelo ER son las entidades, las relaciones, los atributos y las cardinalidades.

- Las diferentes entidades (objetos o sujetos de la base de datos de los que queremos almacenar información) tienen atributos, es decir, propiedades y características, y las entidades se relacionan entre ellas.
- Un atributo o un conjunto de atributos de una entidad puede ser clave primaria, entendida como una propiedad que identifica de forma única un registro de una tabla.
- Por último, se indica la cardinalidad, consistente en la participación que hay entre las entidades.

Cada uno de los elementos del modelo ER se representa de forma distinta. Las entidades se representan mediante rectángulos y sus atributos mediante círculos, por otro lado, las relaciones entre entidades se representan con rombos y, para la cardinalidad de estas, una línea vertical significa "una" y un triángulo "muchas" [22].

La BBDD implementada ha seguido el siguiente modelo ER:

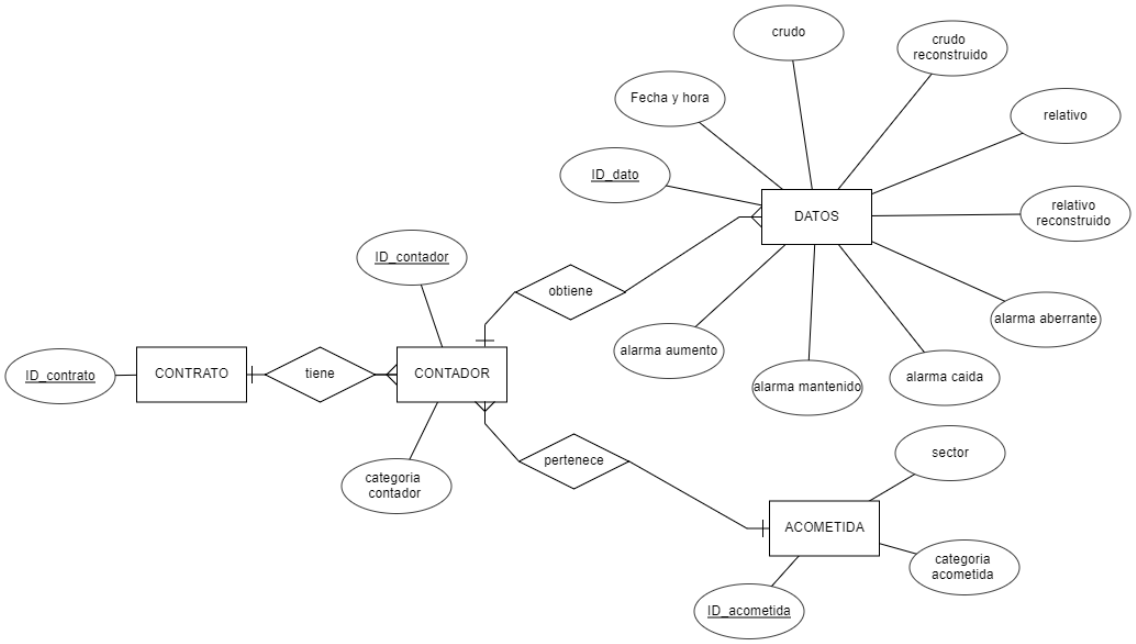


Figura 41.- Modelo ER de la BBDD implementada

La tercera de las fases a seguir para el desarrollo de la BBDD es la transformación del diagrama ER en el modelo relacional.

El modelo relacional es la representación de la base de datos en tablas, donde cada a fila se la denomina tupla y cada columna contiene un atributo distinto. El paso del modelo entidad-relación al modelo relacional varía en función de la cardinalidad general de las relaciones. La siguiente figura representa el diagrama conceptual de este modelo:

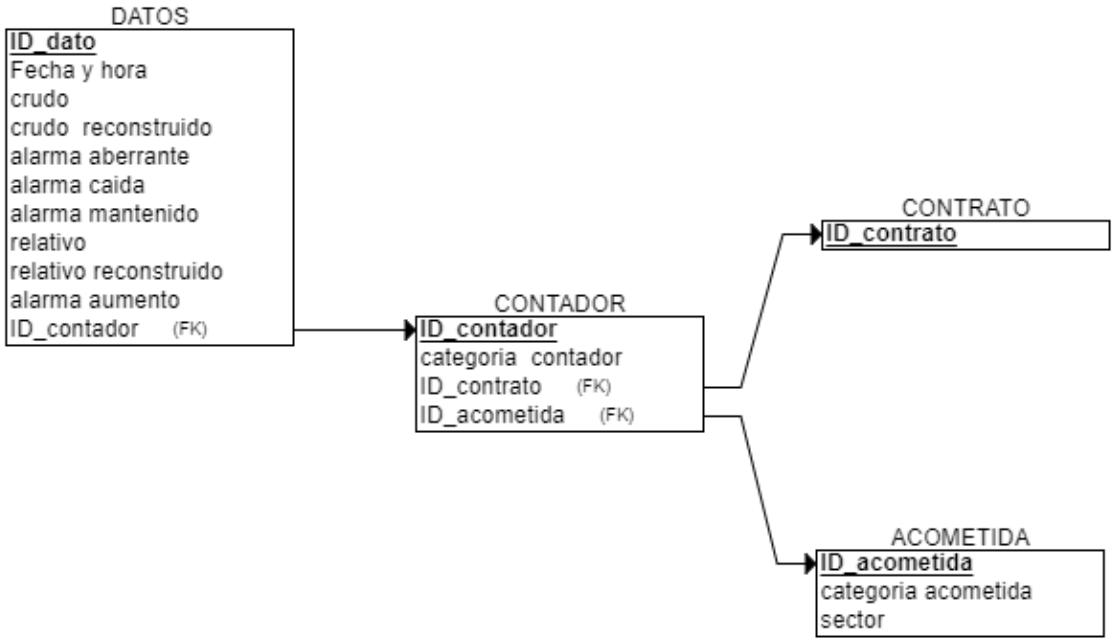


Figura 42.- *Diagrama conceptual del modelo relacional*

En este diagrama nos encontramos con los atributos PK (*Primary Key*) subrayados en negrita y los atributos FK (*Foreign Key*).

- Los atributos PK indican las claves primarias, es decir, identifican de manera única cada entidad, se trata del atributo identificativo de su entidad y, cada entidad tan solo podrá tener uno.
- Los campos marcados como FK son las claves foráneas o claves externas. Éstas indican campos que van a almacenar claves primarias de otras tablas de modo que se puedan relacionar con la tabla actual.

Con todo esto ya tenemos las herramientas para desarrollar el código en SQL que corresponderá a la BBDD.

6.2. Creación y Conexión al Servidor

En el desarrollo de la BBDD se ha dado gran importancia a la facilidad de cualquier usuario para acceder desde cualquier lugar. Esta funcionalidad otorga un valor añadido dentro del mismo diseño de la BBDD, pero para darle esta inmediatez al sistema es necesario utilizar un servidor.

Un servidor de BBDD, también conocido como *database server* o RDBMS (*Relational DataBase Management Systems*) en caso de bases de datos relacionales, es un tipo de software de servidor que permite la organización de la información mediante el uso de tablas, índices y registros [23].

A nivel de hardware, un servidor de BBDD es un equipo informático especializado en servir consultas a clientes remotos o locales que solicitan información o realizan modificaciones a los registros y tablas que existen dentro de las bases de datos del sistema (en muchos casos desde un servidor web o de aplicaciones).

En este caso, el servidor se ha creado mediante la RDBMS de MySQL y se ha consultado con MySQL Workbench.

MySQL es, actualmente, la BBDD de código abierto más popular del mundo tanto para desarrollo web como aplicaciones que utilizan lenguajes de programación como Python, su simplicidad en el uso y la fácil conexión con los scripts que se han desarrollado son los principales motivos por los cuales se ha utilizado esta herramienta.

MySQL cuenta con varias GUI (*Graphical User Interface*), interfaces que permiten a los usuarios interactuar con las BBDD creadas. Pese a que tanto la creación de la BBDD como la alimentación de ésta se realiza mediante Python, MySQL Workbench nos ofrece una GUI donde comprobar que el proceso funciona de forma correcta [24].

La conexión al servidor se realiza mediante librerías de Python, éstas proporcionan funciones capaces de establecer la conexión con el servidor, interactuar (leer y escribir) sobre la RDBMS y gestionar los errores que puedan aparecer a lo largo de la comunicación.

6.3. Creación y Alimentación de la BBDD

SQL ha sido y sigue siendo el lenguaje de programación más usado para la gestión de BBDD, este lenguaje permite la manipulación y descarga de datos de una BBDD explotando la flexibilidad y potencia de los sistemas relacionales.

Se trata de un lenguaje declarativo de alto nivel, es decir, al ejecutar el código desarrollado no se establece explícitamente un orden de ejecución, en los sistemas de BBDD modernos (como es el caso de MySQL), este factor favorece la optimización ya que poseen un componente optimizador que realiza un destallado análisis de los posibles planes de ejecución y escoge el más eficiente.

A nivel programación, los comandos SQL se pueden dividir en cuatro categorías: DDL, DQL, DML y DCL.

- DDL (Data Definition Language): comandos que se pueden utilizar para definir el esquema de la BBDD. Estos comandos suelen ser utilizados por el desarrollador de la BBDD.

- DQL (Data Query Language): se utilizan para realizar consultas sobre los datos dentro de los objetos del esquema. El propósito del comando DQL es obtener una relación de esquema basada en la consulta ejecutada.

- DML (Data Manipulation Language): comandos que se ocupan de la manipulación de los datos presentes en la BBDD. Son los componentes de la sentencia SQL que controlan el acceso a los datos ya la base de datos.

- DCL (Data Control Language): gestionan los derechos, permisos y otros controles del sistema de base de datos [25].

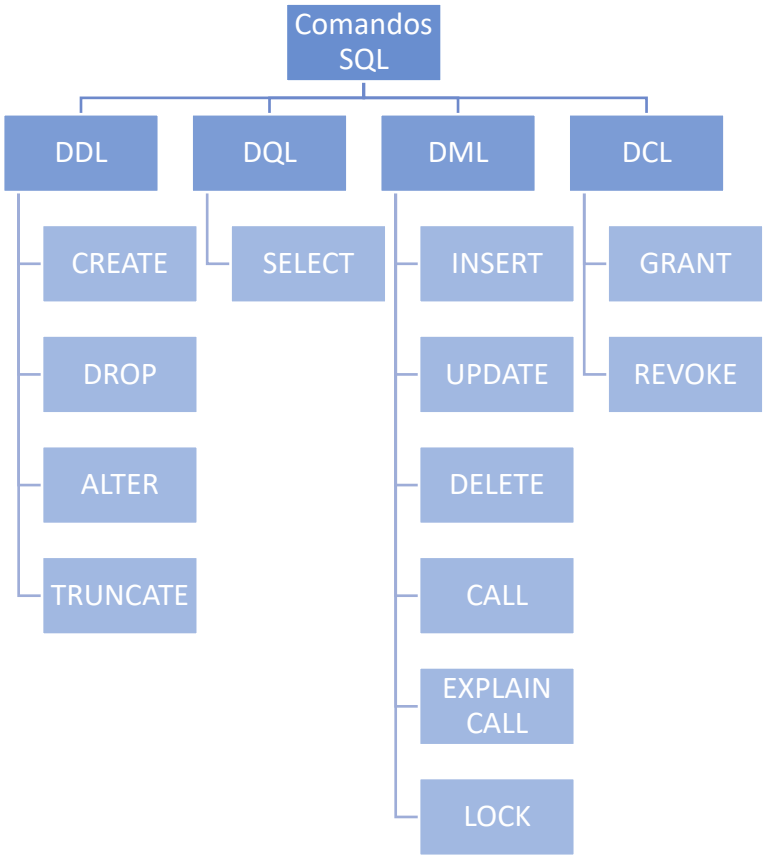
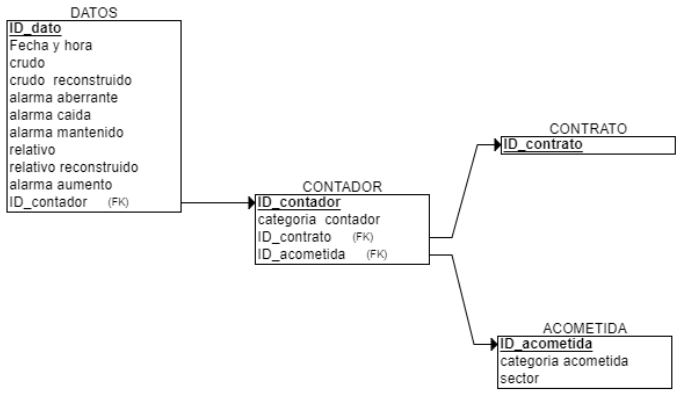


Figura 43.- *Comandos SQL según su categoría*

Para este proyecto se utilizarán comandos de tres de las cuatro categorías del lenguaje SQL; DDL, DQL y DML.

Utilizando el comando CREATE (DDL) y el comando INSERT (DML) será como generaremos la BBDD y la alimentaremos. Las librerías de Python seleccionadas permiten interpretar el lenguaje SQL desde el propio lenguaje de Python, simplificando así ambos procesos.

Si volvemos a los diagramas conceptuales de la BBDD y comparamos con los puntos expuestos en el pretratamiento de datos podemos establecer relaciones, así como entender cuáles son los diferentes parámetros que se introducirán:



	A	B	C
1	hydro_number	DATETIME	Lectura
2	D03XF063055	01/01/2019 3:41	484772010
3	D03XF063055	01/01/2019 4:41	484772120
4	D03XF063055	01/01/2019 5:41	484772240
5	D03XF063055	01/01/2019 6:41	484772350
6	D03XF063055	01/01/2019 7:41	484772470
7	D03XF063055	01/01/2019 8:41	484772590
8	D03XF063055	01/01/2019 9:41	484772700
9	D03XF063055	01/01/2019 10:41	484772820

	A	B	C	D	E	F	G	H	I	J	K
1	code	hydro_number	connec_id	hydrometer_category	cat_hydrometer_id	category_type	sector_id	dma_id	code_2	codi_netaq1	codi_netaq2
2	E1A42175	19BA400882	888	Domestic	Comptador individual	CP	67	67	33864	45547	52701
3	E1T76300	19BE045046	1064	Industrial	Comptador individual	CPI	1	1	5251	1460	20948
4	E1186927	07TB004979	1065	Comercial	Comptador individual	CPI	72	72	5239	111396	141
5	E1185443	11TC148600	1067	Industrial	Comptador individual	CPI	72	72	16405	141	4111
6	E1305593	09TA362836	1068	Comercial	Comptador individual	CPI	72	72	16405	141	4111
7	E1311857	14TA173971	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
8	E1317701	15TA043125	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
9	E1322530	16BA285454	4225	Comercial	Comptador individual	CP	63	63	17165	20008	19744
10	E1312908	14TA155597	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
11	E1312909	14TA135000	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
12	E1312910	14TA134999	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
13	E1313116	14TA155589	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
14	E1313239	14TA135004	10474	Domestic	Comptador individual	ZA-GR	41	41	23405	6189	9055
15	E1316425	15TA015059	10474	Comercial	Comptador individual	ZA-GR	41	41	23405	6189	9055

Figura 44.- *Diagrama conceptual de la BBDD y ejemplos de datos y lecturas de contadores*

- DATOS
    - **ID_dato**: PK de la tabla de datos. Este parámetro será necesario generarlo de cero para cada fila introducida. Para el caso se ha establecido una unión entre el ID del contador y la fecha y hora a la que se ha cogido el dato.
    - **Fecha y hora**: Equivale a la columna "DATETIME" del Archivo de Lecturas (AL).
    - **Crudo**: Equivale a la columna "Lectura" del AL.
    - **Crudo reconstruido**: Valor integral reconstruido.
    - **Relativo**: Valor relativo calculado en el pretratamiento de datos.
    - **Relativo reconstruido**: Valor relativo reconstruido.

- **Alarmas Aberrante, Caída, Mantenido, Aumento**: Alarmas generadas en el apartado de validación de datos.

- CONTADOR
  - **ID_contador**: Equivale a la columna "hydro_number" del Archivo de Referencias (AR).
  - **Categoría contador**: Equivale a la columna "hydrometer_category" del AR.

- CONTRATO
  - **ID_contrato**: Equivale a la columna "code" del AR.
- ACOMETIDA
  - **ID_acometida**: Equivale a la columna "connec_id" del AR.
  - **Categoría acometida**: Equivale a la columna "hydrometer_category" del AR.
  - **Sector**: Equivale a la columna "sector_id" del AR.

Tras tener relacionados todos los parámetros de la BBDD con los datos obtenidos en .CSV, se puede iniciar el proceso de alimentación. Este proceso se ha incluido en el mismo script en el cual se realizaba el pretratamiento, validación y reconstrucción de los datos.

La BBDD completamente alimentada cuenta con 4 tablas, más de 4.7 millones de filas y un total de más de 57 millones de datos introducidos.

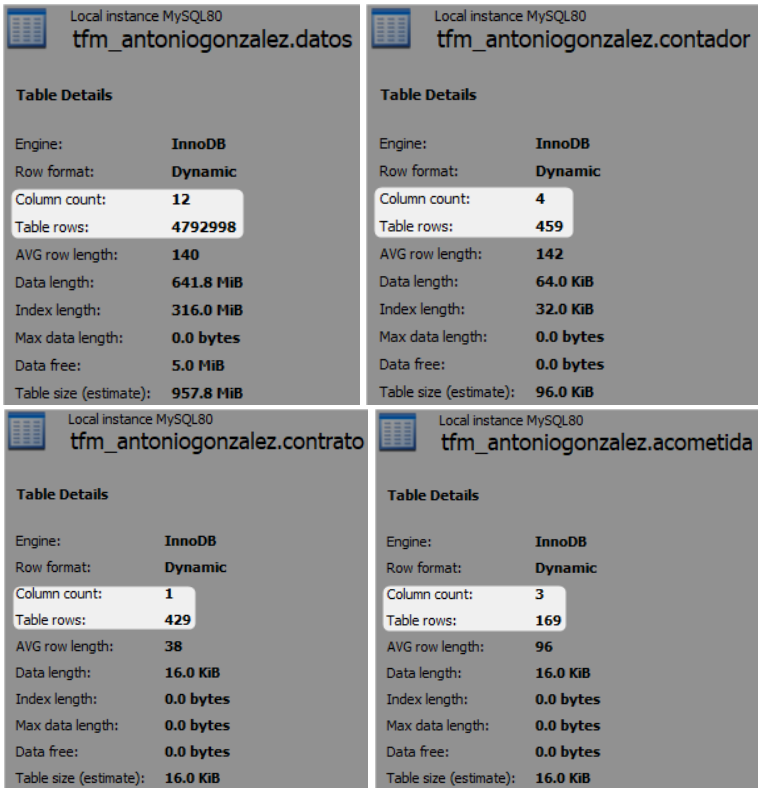


Figura 45.- *Características de las tablas de la BBDD*

La estructura final de cada una de las tablas/entidades de la BBDD resultante son las siguientes:

ID_dato	Fecha_Hora	Valor_Incremental	Valor_Incremental_Reconstruido	Valor_Relativo	Va
01012020000008TA209405	2020-01-01	397944	397939	0	0
01012020000011TA049168	2020-01-01	0	10758300	0	0
01012020000011TB019272	2020-01-01	9727050	9723560	0	0
01012020000014TA173758	2020-01-01	448695	448680	0	0
01012020000014TA173772	2020-01-01	0	546831	0	11
01012020000014TA173819	2020-01-01	412636	412594	17	3
01012020000111TA186416	2020-01-01	5423560	5423270	0	4
01012020000112TA128674	2020-01-01	574473	574433	0	0
01012020000113TA603123	2020-01-01	94607	94600	0	2
01012020000113TA708010	2020-01-01	157541	157482	0	0

Figura 46.- *Muestra (parcial) de las 10 primeras filas de la tabla "datos"*

ID_contador	categoria_contador	ID_contrato	ID_acometida
03XF063055	Industrial	E1193025	14205
04UF081662	Industrial	E1195902	58656
05TA030296	Comercial	E1313684	31425
05TA054664	Comercial	E1177214	16398
05TA054679	Comercial	E1339489	36792
05TA087087	Comercial	E1332664	37090
05TA165870	Comercial	E1198155	16406
05TA199644	Domestico	E1378909	35868
05TA225298	Domestico	E1378071	35868
05TA225335	Comercial	E1377481	41499

Figura 47.- *Muestra de las 10 primeras filas de la tabla "contador"*

ID_contrato
E1153218
E1157412
E1160598
E1161746
E1161761
E1165371
E1166835
E1167376
E1167378
E1170873

Figura 48.- *Muestra de las 10 primeras filas de la tabla "contrato"*

ID_acometida	categoria_acometida	sector
12401	CB	56
11471	CPI	72
11429	CPI	72
10474	ZA-GR	41
4225	CP	63
1068	CPI	72
1067	CPI	72
1065	CPI	72
1064	CPI	1
888	CP	67

Figura 49.- *Muestra de las 10 primeras filas de la tabla "acometida"*

7. CONCLUSIONES

La red de contadores de Terrassa es un elemento crucial dentro de la estructura de distribución del agua de esta ciudad, pero está comenzando a quedarse obsoleta. Actualmente menos del 1% de los contadores de la ciudad cuentan con telemetría, valor muy alejado de otros municipios de Cataluña.

Frente a estos datos encontramos las propuestas por parte del OAT y el despliegue de 220 nuevos telecontadores por toda la ciudad, que avecinan el cambio al que Terrassa está dispuesta a enfrentarse en los próximos años. Con este proyecto se han tratado de establecer unas bases al tratamiento, validación y reconstrucción de los datos procedentes de esta red utilizando mecanismos ETL.

Para el desarrollo del proyecto ha sido necesario en primer lugar disponer de los datos de los contadores actualmente dispuestos por toda la ciudad y pretratarlos para poder interactuar con ellos de una forma más cómoda y eficiente. Los datos (horarios) obtenidos y el estudio realizado sobre los mismos han permitido establecer una serie de tendencias o comportamientos generalmente esperados en el consumo de agua registrado por los contadores, comportamientos los cuales habilitan la introducción de diferentes métodos de validación. Del total inicial de 3.829.859 datos, 1.720.056 de ellos han sido invalidados o puestos bajo estudio para una posterior reconstrucción, en porcentaje, el 44.91% de los datos han sido sometidos a estudio y dispuestos para el proceso de reconstrucción.

La reconstrucción de los datos tiene su gran peso en la implementación de modelos AR dinámicos, los cuales se realimentan con los datos obtenidos externamente o generados para tratar de mantener tendencias de comportamiento en función de cada escenario, en este punto del proyecto contábamos con 4.364.191 datos de lectura finales, de entre los cuales se reconstruyeron 584.172, indicando así que el 13,38% sobre el total de los datos han sido reconstruidos. Estos datos reconstruidos junto a las alarmas disparadas se han introducido en un servidor de BBDD generado para el proyecto, alimentando la estructura y aumentando más aun el alcance de la actividad.

La totalidad del desarrollo se ha realizado utilizando Python como lenguaje de programación y SublimeText3 como entorno de desarrollo, la BBDD se ha realizado utilizando los servidores de MySQL en lenguaje SQL e intérpretes para Python. Con este punto se quiere resaltar que el desarrollo de la actividad en su conjunto no se ha visto limitada a programas de pago o software privativo, por lo tanto, la integridad de la realización de este proyecto no tiene mayor coste que el tiempo y el esfuerzo implicado en su ejecución.

Pese a que el proyecto cumple con el objetivo descrito inicialmente, es interesante evaluar cuáles serían los siguientes pasos a tomar, abrir nuevas líneas de desarrollo y evaluar las posibilidades del sistema. Para ello se han propuesto una serie de ampliaciones y mejoras que han podido ser de interés durante el desarrollo.

7.1. Ampliaciones

El alcance del proyecto abarca un conjunto de requerimientos a cumplimentar para dar por concluido proyecto, entre estos se encontraba principalmente la validación y reconstrucción de datos y la creación y alimentación de la BBDD. Este trabajo pone las bases para posibles desarrollos posteriores que se pueden implementar en todos y cada uno de los apartados tratados.

En este punto se trata de enfocar algunas de las posibles ampliaciones que, por falta de tiempo, no se han realizado pero que podrían haber sido de interés.

7.1.1. Consultas a la BBDD

En el apartado "6. Almacenamiento" se trata la creación y alimentación de la BBDD, pero a la misma no se le da mayor propósito que el almacenamiento constante de los datos crudos, reconstruidos y las alarmas.

El principal uso de las BBDD, al margen del almacenamiento de información, es la capacidad de realizar consultas sobre los datos introducidos mediante sencillos comandos en lenguaje SQL, que permiten acceder de forma rápida y eficiente a sectores muy concretos de la información almacenada.

La ventaja en este punto de haber utilizado Python junto a MySQL es que, la información consultada puede tratarse, generando así la posibilidad de obtener más información aún de la que simplemente nos devolvería una consulta realizada en cualquier otro entorno de consultas.

Para el caso en cuestión se ha generado un script de consultas (véase "8.2. Script de Consultas") que muestra la capacidad de esta ampliación. En el script se recoge la información de los datos reconstruidos para el contador que el usuario desee y, estos datos son mostrados de diversas formas (mediante comportamientos temporales e histogramas), devolviendo también la desviación estándar de los valores relativos de la muestra en concreto.

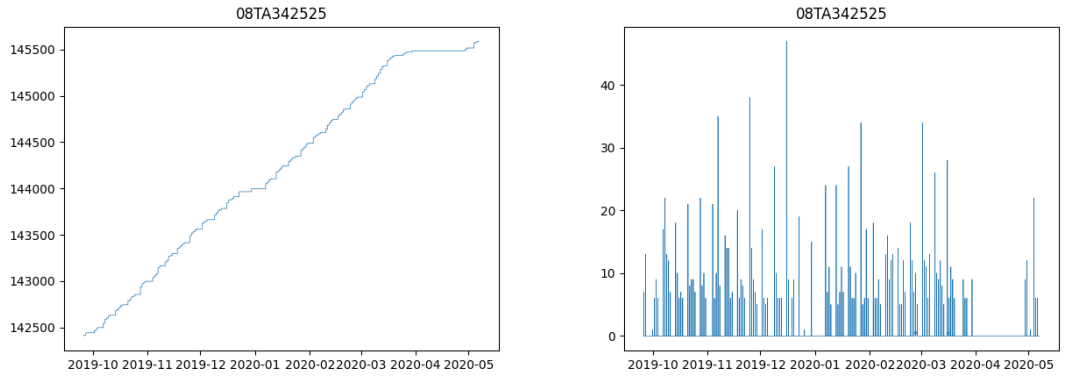


Figura 50.- *(Izda.) Valores incrementales reconstruidos*
(Dcha.) Valores relativos reconstruidos

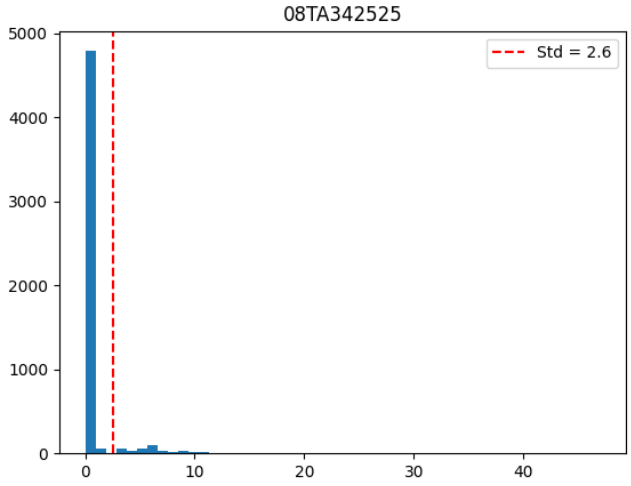


Figura 51.- *Histograma de valores relativos reconstruidos y su desviación estándar.*

En este ejemplo de consulta se han escogido estos parámetros para demostrar que sobre los datos recuperados de la BBDD se puede realizar cualquier tipo de tratamiento, desde la visualización de los datos a cálculos estadísticos complejos pasando por la extracción de únicamente un sector concreto de los datos para exportarlos a documentos de datos (.CSV, .XLSX, …) para darle otros usos.

Mediante esta nueva actividad se podrían crear nuevas herramientas muy potentes que, fundamentalmente, parten del desarrollo inicial de este desarrollo.

7.1.2. Clustering

Los datos proporcionados vienen acompañados de información relativa a los tipos de contadores, tipo de contrato, sector de distribución, etc. Esta información es relevante desde el punto de vista de la reconstrucción ya que, comportamientos similares en diferentes contadores pueden ayudar a tomar como referencia tendencias vistas en contadores diferentes para reconstruir otros datos. Pese al conjunto de información obtenida, pueden existir diversas formas de agrupar contadores, para esto se utilizan las técnicas de *clustering*.

El *clustering* es una tarea (generalmente aplicada en ML) que tiene como finalidad principal lograr el agrupamiento de conjuntos de objetos no etiquetados, para lograr construir subconjuntos de datos conocidos como *clusters*. Cada *cluster* dentro de un grafo está formado por una colección de objetos o datos que a términos de análisis resultan similares entre sí, pero que poseen elementos diferenciales con respecto a otros objetos pertenecientes al conjunto de datos y que pueden conformar un *cluster* independiente. Los criterios para buscar los diferentes *clusters* puede variar en función de los tipos de datos que se están tratando [26].

A modo de ampliación se han generado un conjunto de *clusters* centrado en el CV (Coeficiente de Variación) de los valores relativos reconstruidos de cada contador mediante el algoritmo *k-means*.

En estadística, cuando se desea hacer referencia a la relación entre el tamaño de la media y la variabilidad de la variable, se utiliza el CV. Su fórmula expresa la desviación estándar como porcentaje de la media aritmética, mostrando una interpretación relativa del grado de variabilidad, independiente de la escala de la variable, a diferencia de la desviación típica o estándar [27].

Ecuación 4.- *Cálculo del coeficiente de variación*

$$C_v = \frac{\sigma}{\bar{x}}$$

Por otro lado, el algoritmo *k-means* es un método de agrupamiento, que tiene como objetivo la partición de un conjunto de n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano [28].

Utilizando el CV como criterio de *clustering* y el algoritmo *k-means* (véase "8.3. Script de Clustering") se ha obtenido la siguiente figura:

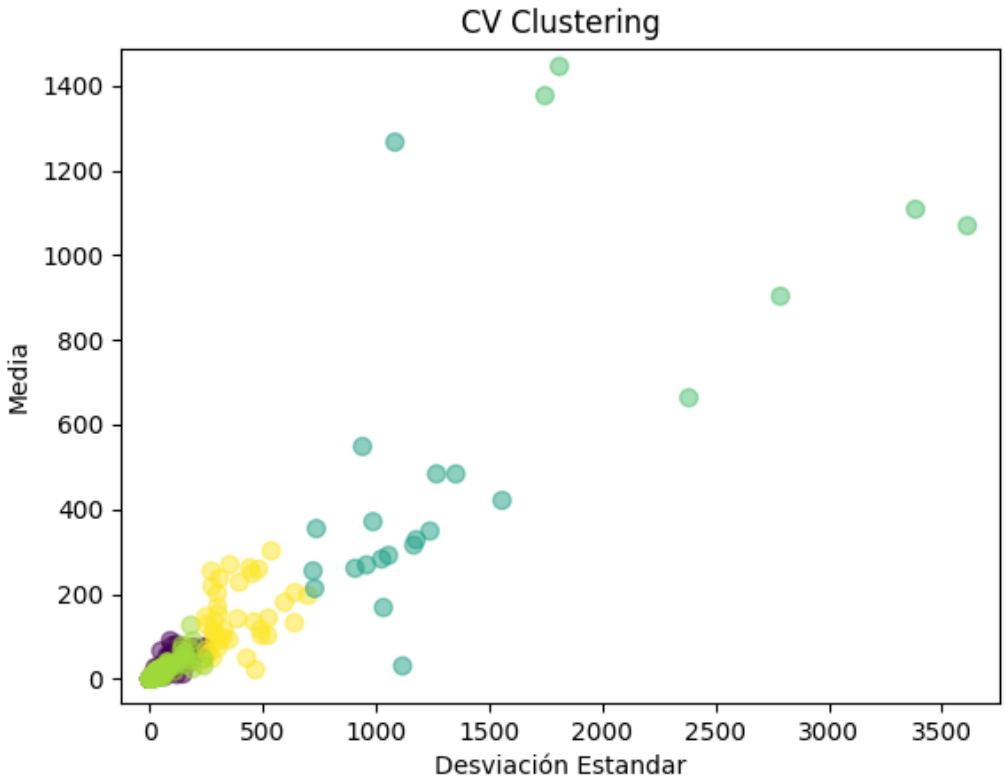


Figura 52.- *Muestra de los diferentes clusters de datos utilizando el CV como criterio*

Pueden existir multitud de métodos de *clustering* mediante los cuales dividir los datos obtenidos por los contadores, no únicamente el CV. Escogiendo un criterio adecuado en función de los *clusters* que se deseen crear, los grupos de contadores resultantes pueden servir para establecer modelos de referencia a la hora de reconstruir los datos.

7.2. Mejoras

A lo largo del desarrollo del proyecto se han tomado acciones que podrían no ser totalmente representativas para el desarrollo de un sistema profesional, con varios ingenieros de software y tiempo suficiente para alcanzar la misma meta que se ha propuesto en este proyecto. En este apartado se tratan de subrayar estos hechos y se dan algunas recomendaciones sobre cómo se debería actuar en el caso de querer realizar un proceso de validación, reconstrucción y almacenamiento como el desarrollado.

Este apartado se centra en dos puntos principales sujetos a mejorar, los cuales son, la reconstrucción de los datos y el mecanismo de ETL utilizado.

7.2.1. Reconstrucción

En el punto "5.1.2. Modelización" se trata el modelo AR dinámico para la reconstrucción de datos en el caso de haber perdido información durante un largo periodo de tiempo o consumo nulo mantenido a lo largo de varios meses.

El principal motivo para utilizar un modelo dinámico para este cometido es el hecho de no tener un marco de datos fijo en las diferentes muestras. Para el desarrollo de esta actividad se ha contado con muestras de año y medio de datos mientras que otras han sido inferiores a los dos meses, esta disparidad en la cantidad de datos por contador fuerza a no poder contar con periodos de aprendizaje concretos.

Los periodos de aprendizaje se han obtenido de la misma muestra que se debía reconstruir, por lo tanto, no existía ningún tipo de apoyo a los datos con muestras de un comportamiento "normal" de, al menos, un año para poder utilizarlos de modelo en el caso de no obtener resultados satisfactorios.

Pese a estas dificultades, más del 90% de los datos contadores reconstruidos (435 de 467) presentan unos valores completamente aceptables, tendencias bien definidas y compatibles con los resultados esperados (ejemplos que ya se vieron en el punto "5.2. Visualización de la Reconstrucción").

Por otro lado, contamos con datos reconstruidos los cuales han sido entrenados con modelos incompatibles a las tendencias esperadas y han dado como resultado valores reconstruidos poco fiables:

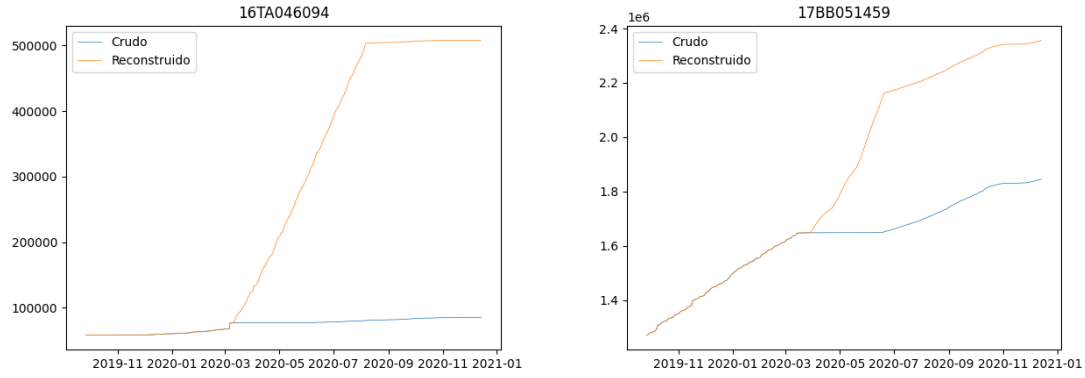


Figura 53.- Ejemplos de lecturas reconstruidos con modelos incompatibles

Para solventar este inconveniente sería preciso tener a disposición cada uno de los contadores modelizados mediante un conjunto de datos, al menos anual, a partir del cual poder reconstruir en el caso de necesitar utilizar el modelo AR.

Para el caso, la mayoría de muestras han quedado bien modelizadas y podrían ser utilizadas con este propósito para posibles lecturas futuras.

7.2.2. ETL y BBDD

La optimización de un mecanismo de ETL pasa por realizar una gestión correcta en cada uno de sus procesos. En nuestro caso, la totalidad del proyecto trata un ETL concreto dónde, se recoge la información de una cantidad concreta de datos en archivos de datos (.CSV), estos datos sufren un proceso de pretratamiento, validación y reconstrucción y, finalmente, son almacenados en una BBDD.

Con las herramientas desarrolladas se ha realizado el proceso completo de forma satisfactoria, pese a que queda el cabo suelto de la próxima facilitación de nuevos datos por parte de TAIGUA.

La primera letra del acrónimo ETL (*Extract*), indica el proceso el cual los datos son dispuestos a análisis. Idealmente estos los formatos de las fuentes se encuentran en BBDD relacionales, pero, en nuestro caso se trata de archivos de datos. Si, estos datos crudos hubiesen sido cargados inicialmente en una BBDD a la cual se nos hubiese permitido el acceso, no sería necesario mantener al sistema a la espera de la próxima recepción masiva de datos, por el contrario, se podrían consultar datos dinámicamente en tiempo real o en periodos de tiempo establecidos de antemano.

Los datos recogidos directamente de la BBDD serían tratados y cargados de nuevo sobre la misma BBDD, introduciendo, en nuestro caso, los parámetros resultantes de la validación (alarmas) y de la reconstrucción (valores incrementales y relativos reconstruidos).

Haber tenido la necesidad de generar una BBDD sobre la cual almacenar los datos es, por un lado, un empeoramiento del propio sistema y una pérdida en la optimización del mecanismo de ETL y, por otro lado, el mal necesario para sentar las bases de una BBDD que nace para almacenar los datos crudos y reconstruidos de todos los contadores la cual se podría seguir alimentando únicamente de los datos crudos por parte del administrador/generador y de los parámetros de alarmas y valores reconstruidos por el algoritmo desarrollado.

8. Anexos

8.1. Script Pretratamiento, Validación, Reconstrucción y Almacenamiento

```
#!/usr/bin/env python

"""Validación, Reconstrucción y Almacenamiento de datos de Telecontadores"""

import pandas as pd [29]
import matplotlib.pyplot as plt [30]
import numpy as np [31]
import datetime
import os
from sklearn.model_selection import train_test_split [32]
from tkinter import Tk, filedialog
from statsmodels.tsa.ar_model import AutoReg [33]
import mysql.connector [34]
from mysql.connector import Error

__author__ = "Antonio Gonzalez Cifuentes"
__credits__ = ["Joseba Quevedo", "Alejandro Matarranz", "Joan Van Eeckhout"]
__version__ = "1.0"
__maintainer__ = "Antonio Gonzalez Cifuentes"
__email__ = "tonigoci97@gmail.com"
__status__ = "Released"

########################
#    PRETRATAMIENTO    #
########################

# Seleccionar las carpetas donde se encuentra la información
root = Tk()
root.withdraw()
root.attributes('-topmost', True)
print('Seleccione la ruta de información de lecturas...',end='\r')
RAW_DATA_FOLDER = filedialog.askdirectory()
print('Seleccione la ruta de información de contadores...',end='\r')
INFO_FOLDER = filedialog.askdirectory()

# RAW_DATA_FOLDER = 'C:/Users/tonig/OneDrive/TFM/DATA/RAW Contadores'
# INFO_FOLDER = 'C:/Users/tonig/OneDrive/TFM/DATA/INFO Contadores'

# Crear dataframes vacíos para rellenar de datos
full_info = pd.DataFrame([])
full_data = pd.DataFrame([])

# Combinar todos los archivos de datos
for file in os.listdir(RAW_DATA_FOLDER):
        if file.endswith('.csv'):
                raw_data = pd.read_csv(f'{RAW_DATA_FOLDER}/{file}')
                full_data = pd.concat([full_data, raw_data])

# Combinar todos los archivos de información
for file in os.listdir(INFO_FOLDER):
        if file.endswith('.csv'):
                raw_info = pd.read_csv(f'{INFO_FOLDER}/{file}', delimiter = ';')
                full_info = pd.concat([full_info, raw_info])


# Añadir datos de csv a dataframe y aplicar la transformacion a datetime
full_data['DATETIME'] = full_data.apply(
        lambda row : datetime.datetime.strptime(row['DATETIME'], '%Y-%m-%d %H:%M:%S'),
        axis = 1)

# Generar diccionario de valores segun el contador ordenando la escala de tiempos
```

```
full_data = full_data.sort_values('DATETIME').groupby(['hydro_number'])

# Iterar por contador y validar individualmente
for id_contador, raw_data_contador in full_data:

	#Se verifica que se tiene información respecto al contador
	if (id_contador in full_info['hydro_number'].unique()):

		raw_data_contador.rename(columns = {'Lectura':'Valor_Incremental'},
			inplace = True)
		raw_data_contador['Valor_Relativo'] =
raw_data_contador['Valor_Incremental'].diff()


		####################
		#    VALIDACIÓN    #
		####################

		'''
		Alarmas
		-------
		Valor perdido -> Lvl 1 - 4
		Caida de valor -> Lvl 1
		Fuera tolerancias -> Lvl 1
		Valor aberrante -> Lvl 1
		Valor mantenido -> Lvl 1 - 4
		Valor aumentado -> Lvl 1 - 4

		'''
		validated_data_contador = raw_data_contador

		# Inicialización de las alarmas
		validated_data_contador['Alarm_Perdida'] =
[0]*len(validated_data_contador['Valor_Incremental'])
		validated_data_contador['Alarm_Caida'] =
[0]*len(validated_data_contador['Valor_Incremental'])
		validated_data_contador['Alarm_Aberrante'] =
[0]*len(validated_data_contador['Valor_Incremental'])
		validated_data_contador['Alarm_Mantenido'] =
[0]*len(validated_data_contador['Valor_Incremental'])
		validated_data_contador['Alarm_Aumento'] =
[0]*len(validated_data_contador['Valor_Incremental'])

		#Establece el indice del dataframe como DATETIME
		validated_data_contador = validated_data_contador.set_index('DATETIME')

		# Generamos dataframe para trabajar las desviaciones estandar y
configurar las alarmas eliminando los ceros
		validated_data_sd = validated_data_contador
		validated_data_sd['Valor_Relativo'] =
validated_data_sd['Valor_Relativo'].mask(validated_data_sd['Valor_Relativo'] < 0, 0)
		validated_data_train, validated_data_test =
train_test_split(validated_data_sd, test_size=0.3)


		# Genera columnas de medias de 90, 30, 15 y 5 días
		days = [90, 30, 15, 5]
		for day in days:
			try:
				Relative_Zeros =
validated_data_contador['Valor_Relativo']
				Relative_Zeros = Relative_Zeros.mask(Relative_Zeros < 0,
0)
				validated_data_contador['Relative Rolling ' + str(day)] =
Relative_Zeros.rolling(str(day)+'D').mean()
			except Exception:
```

```
                        pass

                # Guarda el tiempo inicial de la lista de datos
                initial_time = validated_data_contador['Valor_Incremental'].index[1]

                # Bucle principal de validación
                for idx_validation in
validated_data_contador['Valor_Incremental'].index:

                    #----------ALARMA PÉRDIDA DE DATO----------#
                    # Evalua si entre datos consecutivos ha transcurrido más tiempo
del esperado
                    if (idx_validation - initial_time) > datetime.timedelta(days =
15): validated_data_contador.loc[idx_validation,'Alarm_Perdida'] = 5
                    elif (idx_validation - initial_time) > datetime.timedelta(days =
5): validated_data_contador.loc[idx_validation,'Alarm_Perdida'] = 4
                    elif (idx_validation - initial_time) > datetime.timedelta(days =
1): validated_data_contador.loc[idx_validation,'Alarm_Perdida'] = 3
                    elif (idx_validation - initial_time) > datetime.timedelta(hours
= 5): validated_data_contador.loc[idx_validation,'Alarm_Perdida'] = 2
                    elif (idx_validation - initial_time) > datetime.timedelta(hours
= 1, minutes = 30) and (idx_validation - initial_time) < datetime.timedelta(hours =
2): validated_data_contador.loc[idx_validation,'Alarm_Perdida'] = 1
                    initial_time = idx_validation

                    #---------ALARMA DE CAIDA DE VALOR---------#
                    # Evalua si la diferencia de valor es negativa, lo que significa
que ha habido una caida en la lectura
                    if validated_data_contador['Valor_Relativo'][idx_validation] <
0: validated_data_contador.loc[idx_validation,'Alarm_Caida'] = 1

                    #----------ALARMA VALOR ABERRANTE----------#
                    # Evalua si el valor relativo de la medida está fuera de lo
normal -> 6 veces desviacion std
                    if validated_data_contador['Valor_Relativo'][idx_validation] >
(6*validated_data_train['Valor_Relativo'].std()):
validated_data_contador.loc[idx_validation,'Alarm_Aberrante'] = 1

                    #---------ALARMA DE VALOR MANTENIDO--------#
                    # Evalua si la media en la lectura de datos es baja, por lo
tanto se han estancado los valores
                    if validated_data_contador['Relative Rolling
90'][idx_validation] <= 16*0.1:
validated_data_contador.loc[idx_validation,'Alarm_Mantenido'] = 4
                    elif validated_data_contador['Relative Rolling
30'][idx_validation] <= 16*0.1:
validated_data_contador.loc[idx_validation,'Alarm_Mantenido'] = 3
                    elif validated_data_contador['Relative Rolling
15'][idx_validation] <= 16*0.1:
validated_data_contador.loc[idx_validation,'Alarm_Mantenido'] = 2
                    elif validated_data_contador['Relative Rolling
5'][idx_validation] <= 16*0.1:
validated_data_contador.loc[idx_validation,'Alarm_Mantenido'] = 1

                    #---------ALARMA DE VALOR AUMENTADO--------#
                    if validated_data_contador['Relative Rolling
90'][idx_validation] >= 16*2:
validated_data_contador.loc[idx_validation,'Alarm_Aumento'] = 4
                    elif validated_data_contador['Relative Rolling
30'][idx_validation] >= 16*2:
validated_data_contador.loc[idx_validation,'Alarm_Aumento'] = 3
                    elif validated_data_contador['Relative Rolling
15'][idx_validation] >= 16*2:
validated_data_contador.loc[idx_validation,'Alarm_Aumento'] = 2
```

```
                    elif validated_data_contador['Relative Rolling
5'][idx_validation] >= 16*2:
validated_data_contador.loc[idx_validation,'Alarm_Aumento'] = 1


            # Se limpia el dataframe eliminando las columnas que ya no son
necesarias
            del validated_data_contador['Relative Rolling 5']
            del validated_data_contador['Relative Rolling 15']
            del validated_data_contador['Relative Rolling 30']
            del validated_data_contador['Relative Rolling 90']

            ########################
            #    RECONSTRUCCIÓN    #
            ########################

            # Guardamos los datos obtenidos en una lista para movelos facilmente
            validated_data_contador.reset_index(inplace=True)
            data_list_reconstruct =
validated_data_contador.fillna(0).values.tolist()

            # Iniciamos las listas que se rellenaran de los datos reconstruidos
            raw_absolute = []
            raw_relative = []
            reconstructed_absolute = []
            reconstructed_relative = []
            time = []
            alarm1 = []
            alarm2 = []
            alarm3 = []
            alarm4 = []
            alarm5 = []

            # Bucle principal de reconstrucción
            for i_list,row_list in enumerate(data_list_reconstruct):

                  #---------RECONSTRUCCIÓN DATO PERDIDO--------#
                  # Para periodos cortos se interpola entre los valores extremos
                  if 0 < data_list_reconstruct[i_list][4] <= 3:
                        end_time = data_list_reconstruct[i_list][0]
                        begin_time = data_list_reconstruct[i_list-1][0]
                        steps = int((end_time-begin_time).total_seconds() / 3600)

                        first_value = data_list_reconstruct[i_list-1][2]
                        last_value = data_list_reconstruct[i_list][2]

                        try:
                              incr = (last_value - first_value)/steps
                        except Exception:
                              incr = (last_value - first_value)/1
                        new_relative = first_value

                        # Mientras haya datos perdidos se introducen valores
                        while((end_time - begin_time) > datetime.timedelta(hours
= 1)):

                              begin_time += datetime.timedelta(hours = 1)
                              new_relative += incr

                              reconstructed_relative.append(incr)
                              time.append(begin_time)

                              raw_absolute.append(0)
                              raw_relative.append(0)
                              alarm1.append(data_list_reconstruct[i_list][4])
                              alarm2.append(np.nan)
```

```
                                        alarm3.append(np.nan)
                                        alarm4.append(np.nan)
                                        alarm5.append(np.nan)

                                # Para periodos largos se implementa el AR
                                elif data_list_reconstruct[i_list][4] > 3:
                                        end_time = data_list_reconstruct[i_list][0]
                                        begin_time = data_list_reconstruct[i_list-1][0]

                                        steps = int((end_time - begin_time).total_seconds() /
3600)

                                        # Entrenamiento del modelo
                                        train = reconstructed_relative[-(steps*2):-steps]
                                        model = AutoReg(train, lags=(int(steps/3)))
                                        model_fit = model.fit()

                                        # Se establece el número de datos a introducir
                                        for step in range(steps):
                                                new_relative = 0
                                                for idx,coef in enumerate(model_fit.params):
                                                        if idx == 0:
                                                                new_relative += coef
                                                        else:
                                                                new_relative +=
(coef*reconstructed_relative[-(idx)])
                                                if new_relative < 0 or new_relative > 6 *
np.std(train):

        reconstructed_relative.append(reconstructed_relative[-1])

                                                else:
                                                        reconstructed_relative.append(new_relative)

                                                alarm1.append(data_list_reconstruct[i_list][4])
                                                alarm2.append(np.nan)
                                                alarm3.append(np.nan)
                                                alarm4.append(np.nan)
                                                alarm5.append(np.nan)

                                                raw_absolute.append(0)
                                                raw_relative.append(0)


                                                begin_time += datetime.timedelta(hours = 1)
                                                time.append(begin_time)

                                #---------RECONSTRUCCIÓN CAÍDA VALOR--------#
                                # Se fuerza el valor a 0
                                elif data_list_reconstruct[i_list][5] == 1:
                                        reconstructed_relative.append(0)
                                        time.append(row_list[0])
                                        raw_absolute.append(data_list_reconstruct[i_list][2])
                                        raw_relative.append(data_list_reconstruct[i_list][3])

                                        alarm1.append(data_list_reconstruct[i_list][4])
                                        alarm2.append(data_list_reconstruct[i_list][5])
                                        alarm3.append(data_list_reconstruct[i_list][6])
                                        alarm4.append(data_list_reconstruct[i_list][7])
                                        alarm5.append(data_list_reconstruct[i_list][8])

                                #---------RECONSTRUCCIÓN VALOR ABERRANTE--------#
                                elif 0 < data_list_reconstruct[i_list][4] <= 3:
                                        # Se interpola el dato (si es posible), si no se
introduce la media
                                        try:
```

```
        reconstructed_relative.append((data_list_reconstruct[i_list+1][3] +
data_list_reconstruct[i_list+1][3])/2)
                                except Exception:

        reconstructed_relative.append(data_list_reconstruct['Valor_Incremental'].mean(
))
                                time.append(row_list[0])
                                raw_absolute.append(data_list_reconstruct[i_list][2])
                                raw_relative.append(data_list_reconstruct[i_list][3])

                                alarm1.append(data_list_reconstruct[i_list][4])
                                alarm2.append(data_list_reconstruct[i_list][5])
                                alarm3.append(data_list_reconstruct[i_list][6])
                                alarm4.append(data_list_reconstruct[i_list][7])
                                alarm5.append(data_list_reconstruct[i_list][8])

                        #---------RECONSTRUCCIÓN VALOR MANTENIDO--------#
                        # Se guardan los valores en una lista para trabajarlos
inversamente
                        # para encontrar los momentos en los que aparece un salto de
valor de alarma
                        elif (i_list == len(data_list_reconstruct) or
data_list_reconstruct[i_list][7] < 3) and data_list_reconstruct[i_list-1][7] >= 3:


                                for time_cont, value_alarm in
zip(reversed(time),reversed(alarm4)):

                                        if value_alarm == 3:
                                                begin_time = time_cont
                                                end_time = data_list_reconstruct[i_list-
1][0]
                                        elif value_alarm == 0:
                                                no_alarm_time = time_cont
                                                end_time = data_list_reconstruct[i_list-
1][0]
                                                break;

                                # Si se cumple con el periodo de valor mantenido escogido
se reconstruye con AR
                                if (end_time - no_alarm_time >= datetime.timedelta(days =
45)) and (no_alarm_time - data_list_reconstruct[1][0]  >= datetime.timedelta(days =
45)):

                                        steps = int((end_time-begin_time).total_seconds()
/ 3600)
                                        good_values = int((end_time-
no_alarm_time).total_seconds() / 3600)
                                        train = reconstructed_relative[-(good_values*2):-
good_values]

                                        try:
                                                model = AutoReg(train,
lags=(int(good_values/3)))
                                                del reconstructed_relative[-good_values:]
                                                model_fit = model.fit()

                                                for step in range(good_values):
                                                        new_relative = 0
                                                        for idx,coef in
enumerate(model_fit.params):
                                                                if idx == 0:
                                                                        new_relative += coef
                                                                else:
```

```
                                                        new_relative +=
(coef*reconstructed_relative[-(idx)])
                                                if new_relative < 0 or new_relative
> 6 * np.std(train):
        reconstructed_relative.append(reconstructed_relative[-1])
                                                else:
        reconstructed_relative.append(new_relative)
                                    except Exception:
                                            reconstructed_relative.append(row_list[3])
                                            time.append(row_list[0])
                                            raw_absolute.append(row_list[2])
                                            raw_relative.append(row_list[3])
                                            alarm1.append(row_list[4])
                                            alarm2.append(row_list[5])
                                            alarm3.append(row_list[6])
                                            alarm4.append(row_list[7])
                                            alarm5.append(row_list[8])
                        # Si no hay error se mantienen los valores
                        else:
                                reconstructed_relative.append(row_list[3])
                                time.append(row_list[0])
                                raw_absolute.append(row_list[2])
                                raw_relative.append(row_list[3])
                                alarm1.append(row_list[4])
                                alarm2.append(row_list[5])
                                alarm3.append(row_list[6])
                                alarm4.append(row_list[7])
                                alarm5.append(row_list[8])
                reconstructed_relative = [0] + reconstructed_relative[:-1]
                new_absolute = data_list_reconstruct[0][2]

                for value in reconstructed_relative:
                        new_absolute += value
                        reconstructed_absolute.append(new_absolute)

                # Se guarda el nunevo dataframe
                rebuild_data_contador = pd.DataFrame(
                    {'DATETIME': time,
                    'Valor Incremental': raw_absolute,
                    'Valor Relativo': raw_relative,
                        'Valor Incremental Reconstruido': reconstructed_absolute,
                        'Valor Relativo Reconstruido': reconstructed_relative,
                        'Alarma Dato Perdido': alarm1,
                        'Alarm Caida Valor': alarm2,
                        'Alarm Valor Aberrante': alarm3,
                        'Alarm Valor Mantenido': alarm4,
                        'Alarm Valor Aumento': alarm5
                    })


                ############################
                #    ALIMENTACIÓN BBDD     #
                ############################

                #---------CONNEXION A LA BBDD--------#
                connection = None
                try:
                        connection = mysql.connector.connect(
                                host="localhost",
                                user="root",
```

```
                passwd="************580"
        )
except Error as err:
        pass

#---------CONNEXION A LA DATABASE--------#
cursor = connection.cursor()
try:
        cursor.execute("CREATE DATABASE TFM_antoniogonzalez")
except Error as err:
        pass

connection = None
try:
        connection = mysql.connector.connect(
                host="localhost",
                user="root",
                passwd=""************580"",
                database="TFM_antoniogonzalez"
        )
except Error as err:
        pass

# Estructuración de las tablas
tabla_contrato = """
CREATE TABLE CONTRATO
(
  ID_contrato CHAR(8) NOT NULL,
  PRIMARY KEY (ID_contrato)
);
"""
tabla_acometida = """
CREATE TABLE ACOMETIDA
(
  ID_acometida INT NOT NULL,
  categoria_acometida VARCHAR(10) NOT NULL,
  sector INT NOT NULL,
  PRIMARY KEY (ID_acometida)
);
"""
tabla_contador = """
CREATE TABLE CONTADOR
(
  ID_contador CHAR(10) NOT NULL,
  categoria_contador VARCHAR(20) NOT NULL,
  ID_contrato CHAR(8) NOT NULL,
  ID_acometida INT NOT NULL,
  PRIMARY KEY (ID_contador),
  FOREIGN KEY (ID_contrato) REFERENCES CONTRATO(ID_contrato),
  FOREIGN KEY (ID_acometida) REFERENCES ACOMETIDA(ID_acometida)
);
"""
tabla_datos = """
CREATE TABLE DATOS
(
  ID_dato CHAR(22) NOT NULL,
  Fecha_Hora DATE NOT NULL,
  Valor_Incremental FLOAT NOT NULL,
  Valor_Incremental_Reconstruido FLOAT NOT NULL,
  Valor_Relativo FLOAT NOT NULL,
  Valor_Relativo_Reconstruido FLOAT NOT NULL,
  Alarma_Dato_Perdido INT NOT NULL,
  Alarma_Caida_Valor INT NOT NULL,
  Alarma_Valor_Aberrante INT NOT NULL,
  Alarma_Valor_Mantenido INT NOT NULL,
  Alarma_Valor_Aumento INT NOT NULL,
```

```
                  ID_contador CHAR(10) NOT NULL,
                  PRIMARY KEY (ID_dato),
                  FOREIGN KEY (ID_contador) REFERENCES CONTADOR(ID_contador)
                );
                """

                # Se establece el metodo de alimentación de tablas
                feed_contrato = f"""
                INSERT INTO CONTRATO VALUES
                ("{str(full_info.loc[full_info['hydro_number'] ==
id_contador]['code'].values[0])}");
                """

                feed_acometida = f"""
                INSERT INTO ACOMETIDA VALUES
                ({str(full_info.loc[full_info['hydro_number'] ==
id_contador]['connec_id'].values[0])},\
                "{str(full_info.loc[full_info['hydro_number'] ==
id_contador]['category_type'].values[0])}",\
                {str(full_info.loc[full_info['hydro_number'] ==
id_contador]['sector_id'].values[0])});
                """

                feed_contador = f"""
                INSERT INTO CONTADOR VALUES
                ("{str(id_contador)}",\
                "{str(full_info.loc[full_info['hydro_number'] ==
id_contador]['hydrometer_category'].values[0])}",\
                "{str(full_info.loc[full_info['hydro_number'] ==
id_contador]['code'].values[0])}",\
                {str(full_info.loc[full_info['hydro_number'] ==
id_contador]['connec_id'].values[0])});
                """

                feed_dato = """INSERT INTO DATOS VALUES
                ('"""

                for row_reconstruido in
rebuild_data_contador.fillna(0).values.tolist():
                        feed_dato += str(row_reconstruido[0].strftime("%d%m%Y%H%M") +
id_contador) + "','"
                        feed_dato += str(row_reconstruido[0]) + "',"
                        feed_dato += str(row_reconstruido[1]) + ","
                        feed_dato += str(row_reconstruido[3]) + ","
                        feed_dato += str(row_reconstruido[2]) + ","
                        feed_dato += str(row_reconstruido[4]) + ","
                        feed_dato += str(row_reconstruido[5]) + ","
                        feed_dato += str(row_reconstruido[6]) + ","
                        feed_dato += str(row_reconstruido[7]) + ","
                        feed_dato += str(row_reconstruido[8]) + ","
                        feed_dato += str(row_reconstruido[9]) + ",'"
                        feed_dato += str(id_contador) + "'),\n('"

                feed_dato = feed_dato[:-4] + ';'

                def execute_query(connection, query):
                        cursor = connection.cursor()
                        try:
                                cursor.execute(query)
                                connection.commit()
                        except Error as err:
                                pass

                #---------ALIMENTACIÓN DE TABLAS--------#
                execute_query(connection,tabla_contrato)
                execute_query(connection,tabla_acometida)
```

```
                execute_query(connection,tabla_contador)
                execute_query(connection,tabla_datos)

                execute_query(connection,feed_contrato)
                execute_query(connection,feed_acometida)
                execute_query(connection,feed_contador)
                execute_query(connection,feed_dato)

                print(f'{id_contador} Reconstruido correctamente!\n')

        else:
                pass
```

[35] [36]

8.2. Script de Consultas

```
#!/usr/bin/env python

"""Generación de consulta de BBDD"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime
import os
from sklearn.model_selection import train_test_split
from tkinter import Tk, filedialog
from statsmodels.tsa.ar_model import AutoReg
import mysql.connector
from mysql.connector import Error

__author__ = "Antonio Gonzalez Cifuentes"
__credits__ = ["Joseba Quevedo", "Alejandro Matarranz", "Joan Van Eeckhout"]
__version__ = "1.0"
__maintainer__ = "Antonio Gonzalez Cifuentes"
__email__ = "tonigoci97@gmail.com"
__status__ = "Released"

# Connexión a la BBDD
connection = None
try:
	connection = mysql.connector.connect(
		host="localhost",
		user="root",
		passwd="************580"
	)
except Error as err:
	pass

cursor = connection.cursor()
try:
	cursor.execute("CREATE DATABASE TFM_antoniogonzalez")
except Error as err:
	pass

connection = None
try:
	connection = mysql.connector.connect(
		host="localhost",
		user="root",
		passwd="************580",
		database="TFM_antoniogonzalez"
	)
except Error as err:
	pass


# Generación Consulta 1
# Se recoge el valor relativo reconstruido de un contador concreto
c1 = """
SELECT Fecha_Hora,Valor_Relativo_Reconstruido
FROM tfm_antoniogonzalez.datos
where ID_contador = '08TA342525'
order by Fecha_Hora;
"""

# Generación Consulta 1
# Se recoge el valor incremental reconstruido de un contador concreto
c2 = """
SELECT Fecha_Hora,Valor_Incremental_Reconstruido
```

```
FROM tfm_antoniogonzalez.datos
where ID_contador = '08TA342525'
order by Fecha_Hora;
"""

# Función de ejecución de lecturas
def read_query(connection, query):
        cursor = connection.cursor()
        result = None
        try:
            cursor.execute(query)
            result = cursor.fetchall()
            return result
        except Error as err:
                print(f"Error: '{err}'")

consulta1 = read_query(connection, c1)
from_db = []
for datos in consulta1:
        datos = list(datos)
        from_db.append(datos)
columns = ["DATETIME", "Relativo"]
df_c1 = pd.DataFrame(from_db, columns=columns)

# Muestreo del valor relativo
plt.plot(df_c1['DATETIME'],df_c1['Relativo'], linewidth=0.5)
plt.title('08TA342525')
plt.show()
plt.clf()

# Muestreo de la desviación estandar del valor relativo
plt.hist(df_c1['Relativo'],bins=50)
plt.axvline(x = df_c1['Relativo'].std(),ls='--',color='red',label=(f'Std =
{round(df_c1["Relativo"].std(),2)}'))
plt.title('08TA342525')
plt.legend()
plt.show()
plt.clf()

consulta2 = read_query(connection, c2)
from_db = []
for datos in consulta2:
        datos = list(datos)
        from_db.append(datos)

# Muestreo del valor Absoluto
columns = ["DATETIME", "Absoluto"]
df_c1 = pd.DataFrame(from_db, columns=columns)
plt.plot(df_c1['DATETIME'],df_c1['Absoluto'], linewidth=0.5)
plt.title('08TA342525')
plt.show()
```

8.3. Script de Clustering

```
#!/usr/bin/env python

"""Clustering de datos segun CV"""

""" Para ejecutar este script es necesario tener los datos de
todos los contadores guardados en archivos con formato XLSX"""

import datetime
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
from sklearn.cluster import KMeans

__author__ = "Antonio Gonzalez Cifuentes"
__credits__ = ["Joseba Quevedo", "Alejandro Matarranz", "Joan Van Eeckhout"]
__version__ = "1.0"
__maintainer__ = "Antonio Gonzalez Cifuentes"
__email__ = "tonigoci97@gmail.com"
__status__ = "Released"

# Selección de datos e inicialización de variables
root = Tk()
root.withdraw()
root.attributes('-topmost', True)
FOLDER = filedialog.askdirectory()

cluster_data = pd.DataFrame({'Std':[],
                                                    'Mean':[]})

# Recogida de la media y desviación estandar de los datos relativos reconstruidos
for file in os.listdir(FOLDER):
        if file.endswith('.xlsx'):
                new_df = pd.read_excel(FOLDER + file)
                new_row = pd.DataFrame({'Std':[new_df['Relative_Reconstr'].std()],

        'Mean':[new_df['Relative_Reconstr'].mean()]})
                cluster_data = pd.concat([cluster_data,new_row], ignore_index=True)
        print(file[:-5])

# Generación de clusters de datos
kmeans = KMeans(n_clusters=8).fit(cluster_data)
centroids = kmeans.cluster_centers_
print(centroids)

plt.scatter(cluster_data['Std'], cluster_data['Mean'], c=
kmeans.labels_.astype(float), s=50, alpha=0.5)
plt.title('CV Clustering')
plt.xlabel('Desviación Estandar')
plt.ylabel('Media')
plt.show()
```

9. Bibliografía

[1] «Smart Water Metering Market by Type, Technology, Component, Application and Region - Global Forecast to 2024,» Markets&Markets, 2019.

[2] «Moció de L'OAT al ple de l'ajuntament de Terrassa» Observatori de l'Aigua de Terrassa, 28 Enero 2021. [En línea]. Available: https://www.oat.cat/Oat/wp-content/uploads/2020/05/Punt-6.-Moció-OAT.pdf. [Último acceso: 1 Mayo 2022].

[3] «Historia» Mina serveis d'aigua, 13 Agosto 2020. [En línea]. Available: https://www.minaserveisdaigua.com/es/historia-2/. [Último acceso: 11 Mayo 2022].

[4] «Quiénes somos» Taigua, 4 Agosto 2020. [En línea]. Available: https://taigua.cat/es/quienes-somos/. [Último acceso: 11 Mayo 2022].

[5] «Telelectura - Web oficial - La gestió responsable» Aigües de Barcelona, 3 Junio 2021. [En línea]. Available: https://www.aiguesdebarcelona.cat/es/tu-servicio-de-agua/mi-factura-consumo/telelectura. [Último acceso: 1 Mayo 2022].

[6] «Data conditioning» Wikipedia, 1 Octubre 2009. [En línea]. Available: https://en.wikipedia.org/wiki/Data_conditioning#:~:text=Data%20conditioning%20is%20the%20use,movement%20in%20a%20computer%20system. [Último acceso: 2 Marzo 2022].

[7] «Redes de estaciones meteorológicas automáticas: Directrices para la validación de registros meteorológicos procedentes de redes de estaciones automáticas» AENOR UNE 500540, 2004.

[8] «Valor atípico» Wikipedia, 9 Marzo 2008. [En línea]. Available: https://es.wikipedia.org/wiki/Valor_atípico. [Último acceso: 9 Mayo 2022].

[9] «Quartile» Wikipedia, 7 Febrero 2004. [En línea]. Available: https://en.wikipedia.org/wiki/Quartile. [Último acceso: 5 Abril 2022].

[10] «Interquartile range» Wikipedia, 23 Julio 2004. [En línea]. Available: https://en.wikipedia.org/wiki/Interquartile_range. [Último acceso: 9 Abril 2022].

[11] «Desviación típica» Wikipedia, 23 Septiembre 2003. [En línea]. Available: https://es.wikipedia.org/wiki/Desviación_típica. [Último acceso: 2022 Marzo 2022].

[12] «Regla 68-95-99.7» Wikipedia, 22 Abril 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Regla_68-95-99.7. [Último acceso: 22 Marzo 2022].

[13] «Half-normal distribution» Wikipedia, 23 Febrero 2008. [En línea]. Available: https://en.wikipedia.org/wiki/Half-normal_distribution. [Último acceso: 9 Abril 2022].

[14] «Folded normal distribution» Wikipedia, 4 Agosto 2006. [En línea]. Available: https://en.wikipedia.org/wiki/Folded_normal_distribution. [Último acceso: 10 Abril 2022].

[15] «Moving average» Wikipedia, 16 Septiembre 2004. [En línea]. Available: https://en.wikipedia.org/wiki/Moving_average. [Último acceso: 9 Mayo 2022].

[16] «Modelado Numérico» Wikipedia, 14 Abril 2008. [En línea]. Available: https://es.wikipedia.org/wiki/Modelado_numérico. [Último acceso: 5 Abril 2022].

[17] «Autoregressive (AR) models with Python» Data Analytics, 21 Marzo 2021. [En línea]. Available: https://vitalflux.com/autoregressive-ar-models-with-python-examples/#:~:text=Autoregressive%20(AR)%20models%20are%20a,order%20to%20make%20accurate%20predictions.. [Último acceso: 20 Mayo 2022].

[18] «Hard-coding» Wikipedia, 13 Septiembre 2006. [En línea]. Available: https://en.wikipedia.org/wiki/Hard_coding. [Último acceso: 15 Mayo 2022].

[19] «Extract, transform and load» Wikipedia, 9 Enero 2010. [En línea]. Available: https://es.wikipedia.org/wiki/Extract,_transform_and_load. [Último acceso: 25 Febrero 2022].

[20] «Database» Wikipedia, 6 Enero 2004. [En línea]. Available: https://en.wikipedia.org/wiki/Database. [Último acceso: 5 Abril 2022].

[21] «SQL» Wikipedia, 17 Enero 2003. [En línea]. Available: https://en.wikipedia.org/wiki/SQL. [Último acceso: 26 Abril 2022].

[22] «El modelo Entidad-Relación: el esquema de una base de datos» ILERNA , 12 Noviembre 2019. [En línea]. Available: https://www.ilerna.es/blog/informatica-comunicacion/modelo-entidad-relacion-base-de-datos/. [Último acceso: 25 Mayo 2022].

[23] «RDBMS» Wikipedia, 23 Junio 2011. [En línea]. Available: https://es.wikipedia.org/wiki/Sistema_de_gestión_de_bases_de_datos_relacionales. [Último acceso: 31 Marzo 2022].

[24] «Workbench» MySQL, 3 Junio 2008. [En línea]. Available: https://dev.mysql.com/doc/workbench/en/. [Último acceso: 26 Mayo 2022].

[25] «SQL | DDL, DQL, DML, DCL and TCL Commands» GeeksforGeeks , 6 Noviembre 2017. [En línea]. Available: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/. [Último acceso: 26 Mayo 2022].

[26] «Cluster Analisys» Wikipedia, 15 Diciembre 2004. [En línea]. Available: https://en.wikipedia.org/wiki/Cluster_analysis. [Último acceso: 3 Junio 2022].

[27] «Coefficient of variation» Wikipedia, 30 Abril 2005. [En línea]. Available: https://en.wikipedia.org/wiki/Coefficient_of_variation. [Último acceso: 28 Mayo 2022].

[28] «k-means clustering» Wikipedia, 15 Marzo 2010. [En línea]. Available: https://en.wikipedia.org/wiki/K-means_clustering. [Último acceso: 6 Junio 2022].

[29] «Pandas Documentation» pandas - Python Data Analysis Library, 5 Febrero 2020. [En línea]. Available: https://pandas.pydata.org/docs/index.html. [Último acceso: 25

Febrero 2022].

[30] «Visualization with Python» Matplotlib, 30 Agosto 2012. [En línea]. Available: https://matplotlib.org. [Último acceso: 6 Marzo 2022].

[31] «NumPy v1.22 Manual» NumPy, 12 Julio 2020. [En línea]. Available: https://numpy.org/doc/stable/user/absolute_beginners.html. [Último acceso: 6 Marzo 2022].

[32] «Stack Overflow» Stack Overflow, 15 Octubre 2013. [En línea]. Available: https://scikit-learn.org. [Último acceso: 29 Mayo 2022].

[33] «Autoregressive AR-X(p) model» Statsmodels 0.14.0, 6 Mayo 2020. [En línea]. Available: https://www.statsmodels.org/devel/generated/statsmodels.tsa.ar_model.AutoReg.html. [Último acceso: 1 Mayo 2022].

[34] «MySQL Connecto Documentation» MySQL, 12 Septiembre 2012. [En línea]. Available: https://dev.mysql.com/doc/connector-python/en/connector-python-examples.html. [Último acceso: 10 Mayo 2022].

[35] «Stack Overflow» Stack Overflow, 2015 Diciembre 2015. [En línea]. Available: https://es.stackoverflow.com. [Último acceso: 29 Mayo 2022].

[36] «Index of Python Enhancement Propostal (PEPs)» PEP 8 - Style Guide for Python Code, 6 Noviembre 2017. [En línea]. Available: https://peps.python.org/pep-0008/. [Último acceso: 27 Mayo 2022].