Universidad Zaragoza
1542

Trabajo Fin de Grado

Modelos para la detección de ataques en redes inteligentes: Desarrollo, evaluación y optimización

Models for the detection of attacks in smart grids: Development, evaluation and optimization

Autor

Guillermo Cánovas González

Directores

José Javier Merseguer Hernáiz

Simona Bernardi

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2023

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a José y a Simona ya que ha sido una experiencia enormemente gratificante y enriquecedora poder trabajar junto a vosotros. Desde el primer día habéis hecho que todo fuera muy fácil y desde entonces todo ha ido sobre ruedas. Sois unos docentes y unas personas excelentes.

Por supuesto, gracias a mis amigos, que me han acompañado a lo largo de este duro trayecto, especialmente a Jordan, Sergio y Huszak. Estoy muy contento de haber podido compartir todo esto con vosotros, gracias por todos los recuerdos que guardamos y gracias porque siempre se puede contar con vosotros para todo.

Gracias a mi familia, en especial a mi madre, a mi tía y a Arti. Aunque muchas veces me cueste expresar lo que siento, estoy eternamente agradecido con todos vosotros por todo el apoyo recibido a lo largo de estos años de carrera. También, gracias a Charo y a Luis, que aunque no seáis familia, habéis estado siempre los primeros para alegraros por mis logros como si fuesen vuestros.

Por último, gracias Sara, gracias por haber estado estos dos últimos años a mi lado, has estado en los momentos buenos, en los malos y en los peores. Gracias por confiar en mí cuando ni yo mismo lo hacía, por empujarme siempre a seguir adelante, por haberme enseñado tantísimo y por hacerme mejor persona día tras día. Estoy convencido de que sin ti, nada de esto sería posible. Por todo esto y mucho más, gracias. Dentro de menos de lo que crees serás la mejor ingeniera de todas.

Modelos para la detección de ataques en redes inteligentes: Desarrollo, evaluación y optimización

RESUMEN

Las redes inteligentes significan un gran avance en la gestión y distribución de la energía eléctrica, gracias al uso de diferentes tecnologías y sensores, estas son capaces de adaptarse mejor a la oferta y a la demanda de energía en tiempo real. El resultado de la implantación de este tipo de redes es una gestión más eficiente de la energía, que beneficia tanto al medio ambiente como a la sociedad, garantizando suministros de electricidad más estables y confiables.

Sin embargo, como toda evolución, esta trae consigo nuevos desafíos, especialmente en lo que respecta a la seguridad, ya que, al integrarse en sistemas tecnológicos, la fiabilidad de los datos puede verse comprometida por múltiples tipos de ciberataques. Estos suponen un beneficio económico para el atacante pero repercuten negativamente tanto en los proveedores, como en los consumidores de esta energía. Entre estas consecuencias se encuentran la posible interrupción del suministro energético, la pérdida de confianza en la red y el aumento en los precios de las tarifas.

En este contexto, el presente trabajo se enfoca en: diseñar, desarrollar, evaluar y optimizar modelos que sean capaces de identificar fraudes en las redes inteligentes o *smart grids*.

Mediante la utilización de datos reales del consumo de electricidad, gas y energía solar, estos modelos, basados en técnicas estadísticas o de aprendizaje automático, tienen como objetivo buscar patrones anómalos que puedan indicar un posible ataque.

Índice

Capítulo 1

Introducción

Este capítulo tiene como objetivo, en primer lugar, contextualizar el marco y los motivos que llevaron a realizar el proyecto (Sección 1.1), en segundo lugar, detallar los objetivos y el alcance del mismo (Secciones 1.2 y 1.3) y, por último, describir la estructura del documento que servirá para conocer como se ha organizado la información (Sección 1.4).

1.1. Contexto

Las redes inteligentes (*smart grids*), tal como se ha señalado en el resumen, al hacer uso de diferentes recursos tecnológicos, suponen un gran avance en la gestión y distribución de la energía eléctrica; sin embargo, pueden verse afectadas por diferentes tipos de amenazas, lo que conlleva la aparición de nuevos retos relacionados con las mismas.

En este proyecto las amenazas consideradas son los fraudes a operadores de redes inteligentes mediante la falsificación de los registros de los contadores inteligentes, es decir, la manipulación de los datos de consumo reales. Para la detección de estos posibles ciberataques se han propuesto distintas soluciones, en particular, modelos basados en técnicas estadísticas o de aprendizaje automático [1].

El punto de partida de este trabajo, consiste en una serie de resultados obtenidos a partir de la experimentación de un grupo de modelos de detección [2] sobre dos conjuntos de datos que provienen del *Comision for Energy Regulation (CER)*, concretamente del *Irish Social Science Data Archive* (*ISSDA*) [3]. Si bien estos resultados sirven como una visión inicial acerca de la eficacia de los detectores ante diferentes escenarios de consumo anómalo, es decir, cuáles son capaces de detectar mejor un tipo de ataque u otro, también indican ciertas áreas de mejora que serán las que guíen la dirección de nuestro trabajo.

Los modelos evaluados previamente se basan en técnicas estadísticas o de

aprendizaje automático. Estos demostraron ser efectivos en muchos escenarios, sin embargo, se podía observar como prácticamente todos los detectores, dependiendo de sus características concretas, eran capaces de detectar con una precisión muy alta algunos tipos de ataques, mientras que eran incapaces de detectar algunos otros.

Un ejemplo de esto son los detectores *KLD* y *JSD*, que son capaces de predecir con una precisión superior al 90 % los ataques *FDI10* y *Avg*, pero no consiguen detectar más de un 20 % de los ataques totales de los tipos *RSA 0.25 1.1* y *Swap*. Los diferentes detectores, así como los escenarios estudiados, se detallan en el Capítulo 2.

1.2. Objetivos y enfoque metodológico

En esta sección, a partir de los resultados comentados en la Sección 1.1, se definen y describen los objetivos principales del trabajo presente:

- Reproducir los resultados previos
- Evaluar los detectores con un nuevo conjunto de datos
- Evaluar los detectores *teg*
- Buscar la forma de mejorar la precisión de ciertos detectores
- Implementar nuevos modelo que puedan ofrecer una detección mejor que los anteriores

El primero de los objetivos del proyecto, fue garantizar la fiabilidad de los resultados de la experimentación de [2] para asegurar que estos no fueron producto de variables no consideradas. Completar este objetivo fue relativamente sencillo, simplemente se repitieron los experimentos utilizando exactamente los mismos datos y detectores y se compararon estos resultados con los del trabajo de referencia.

Con la confianza en los resultados iniciales, se pasó al siguiente objetivo. Se probaron los mismos detectores, pero esta vez utilizando un nuevo conjunto de datos con la intención de verificar su consistencia en datos de redes inteligentes diferentes.

Asimismo, gracias a la incorporación de este *dataset* basado en la energía solar [4], se introdujo una nueva variante de ataques. Debido a que en las redes inteligentes existe la opción del aporte energético vía paneles solares, se aprovechó para crear escenarios de ataque que consisten en declarar más cantidad de energía de la que realmente se está aportando a la red. Es decir, a diferencia de los anteriores, no se enfocan en el consumo sino en la generación.

El siguiente punto abordado fue la evaluación de los detectores *teg* [5] para comprobar su capacidad de detección y compararla con el resto de modelos.

Además, otro de los objetivos marcados era la mejora del modelo *NN*, basado en redes neuronales. Esto se debe a que en la experimentación realizada en [2], este demuestra ser el mejor a la hora de detectar consistentemente todos los tipos de ataques con una precisión considerablemente alta. Pese a esto, había ciertos aspectos que requerían ser estudiados para poder aprovechar al máximo la red neuronal. Uno de estos aspectos fue el *overfitting*, que, a causa de este, el modelo presentaba una elevada cantidad de falsos positivos que le impedía detectar correctamente aquellos datos que no habían sido manipulados.

Finalmente, con la ambición de ir más allá, se tomó la decisión de diseñar y probar un modelo de detección más avanzado. Utilizando redes neuronales recurrentes *LSTM (Long Short-Term Memory)* [6], se buscó determinar si un enfoque más complejo podría obtener mejores resultados en la detección de los escenarios.

Toda la experimentación comentada ha sido llevada a cabo en el *clúster* de supercomputación *"Hermes"* del *I3A*, que cuenta con 8 núcleos, un procesador Intel Xeon Gold 6254 CPU, 32 GB de RAM y 1 TB de almacenamiento SSD.

1.3. Alcance del proyecto

El objetivo de este apartado es definir el alcance del trabajo, es decir, lo que se ha intentado abordar y lo que ha sido excluido de manera intencional.

El proyecto se centra principalmente en la evaluación, desarrollo y optimización de modelos para la detección de ataques. Estos detectores han sido probados utilizando distintos conjuntos de datos, pero todos ellos dentro de escenarios controlados, es decir, pese a que ciertos modelos podrían adaptarse para realizar detecciones en tiempo real, este aspecto no ha sido considerado.

Por último, es importante mencionar que, un aspecto muy importante al trabajar con datos que puedan contener información personal, es el tema de la privacidad. Sin embargo, esto no se ha tenido en cuenta ya que los datos utilizados en este proyecto provienen de conjunto de datos abiertos y anonimizados.

1.4. Estructura del documento

Inicialmente, el Capítulo 2 se utiliza para explicar las bases de la experimentación, es decir, los conjuntos de datos empleados y los distintos escenarios considerados. Después, en el Capítulo 3 se detallan los distintos modelos de detección, tanto los que se han implementado como aquellos que ya lo estaban pero que han sido nuevamente evaluados. Posteriormente, en el Capítulo 4 se describe brevemente la experimentación,

se resumen sus distintas fases y algunos de los resultados obtenidos. Seguidamente, el Capítulo 5, explica la metodología de trabajo utilizada, la planificación que se ha seguido y las horas totales de dedicación al proyecto. La última parte de la memoria es el Capítulo 6, en donde se presenta las conclusiones que se extraen de la realización del proyecto, se comentan los objetivos logrados, las lecciones aprendidas y las posibles futuras mejoras.

Finalmente, con intención de incluir aquella información adicional relevante, pero que no se ha añadido en el cuerpo de la memoria para mejorar la comprensión de la misma, se emplean 6 anexos. El Anexo A sirve para destacar ciertos módulos que pueden resultar de interés y también sirve de guía para ejecutar ciertas tareas que necesitan de una breve explicación previa. En el Anexo B se describen con mayor detalle los conjuntos de datos utilizados. A continuación, en el Anexo C se listan cada una de las herramientas utilizadas. El Anexo D se usa para presentar visualmente la planificación de tareas a lo largo del proyecto y el tiempo total dedicado al mismo. Después de este, el Anexo E explica las distintas métricas empleadas en la evaluación y comparación de detectores. Por último, el más extenso de todos, el Anexo F se encarga de recoger todas las tablas de resultados del proceso de experimentación.

Capítulo 2

Bases de la experimentación

En este capítulo se explican los pilares fundamentales de la experimentación: los conjuntos de datos utilizados (Sección 2.1) y los diferentes escenarios de consumo y de generación que se han considerado (Secciones 2.2 y 2.3).

2.1. Conjuntos de datos

Esta sección tiene como objetivo explicar los distintos *datasets* empleados en la experimentación del trabajo (electricidad, gas y energía solar). Es importante señalar que, independientemente del conjunto de datos utilizado, los datos finales con los que se trabajan son los mismos: valores de consumo energético en kilovatio hora registrados por cada contador cada 30 minutos durante cierto número de semanas que depende del conjunto en cuestión.

2.1.1. Electricidad y gas (ISSDA CER)

Para la realización del trabajo, tal y como se hizo en [2], se han elegido dos conjuntos que contienen datos provenientes de una red inteligente. Concretamente, se han empleado dos conjuntos de datos proporcionados por la *Comission for Energy Regulation* (*CER*) y obtenidos específicamente del *Irish Social Science Data Archive* (*ISSDA*) [3]. Uno de ellos recopila información sobre el consumo de electricidad y el otro sobre el consumo de gas.

En lo que respecta al conjunto de la electricidad, este recopila información sobre el consumo de 6445 contadores en Irlanda. Estos registran datos de consumo (en kilovatio hora) cada 30 minutos durante 75 semanas. Por otro lado, el conjunto de datos del gas recoge la misma información, pero, en este caso, sobre el consumo de gas de 1576 contadores durante 77 semanas, también en Irlanda. Estos se describen con mayor detalle en los Anexos B.1 y B.2.

2.1.2. Energía solar (Ausgrid)

El conjunto de datos utilizado para la energía solar es *Ausgrid Solar Dataset* [4] que proviene de *Ausgrid*, una de las principales empresas energéticas en Australia. Este *dataset* recopila información sobre el consumo y la generación de energía solar (en kilovatio hora) cada 30 minutos durante 101 semanas de diferentes sistemas de paneles solares en viviendas y establecimiento de *New South Wales*, Australia. Los datos recogidos se pueden observar con detalle en la Tabla B.3.

Un aspecto que diferencia a este, respecto a los dos anteriores, es la inclusión para cada contador no solo de datos de consumo, sino también de generación. Esto fue esencial tenerlo en cuenta durante el preprocesado, donde se dividieron los datos en dos bloques: generación y consumo. Se puede consultar más información sobre este proceso en el Anexo B.3.

2.2. Escenarios de consumo

Se van a distinguir dos tipos de escenarios, por un lado, el escenario normal, que representa el consumo registrado por un contador sin ningún tipo de manipulación y, por otro lado, los diferentes escenarios de ataque que se van a estudiar, en estos se manipulan intencionalmente los valores de los registros con el objetivo de obtener un beneficio económico.

2.2.1. Normal

En este escenario no se realiza ningún tipo de modificación en los datos registrados por el contador. Es importante señalar que, pese a que las muestras no se manipulan, puede presentar ciertas anomalías puntuales. Esto se debe a situaciones atípicas de la vida de las personas, por ejemplo, unas vacaciones, ya que durante ese período de tiempo el contador no va a ser utilizado. Los detectores han de ser capaces de no percibir estos comportamientos como anómalos o de lo contrario presentarán un alto número de falsos positivos.

2.2.2. False Data Injection

El ataque *False Data Injection* (FDI) consiste en multiplicar todos los registros por un valor constante X. Dentro de este escenario se consideran dos variantes: FDI 10 y FDI 30 que corresponden a declarar un 10 % y 30 % del consumo real, respectivamente, o lo que es lo mismo, un ahorro por parte del atacante del 90 % y 70 % del total de la factura.

2.2.3. Random Scale Attack

En el escenario *Random Scale Attack* [7] (RSA), los valores de consumo se multiplican por un valor aleatorio dentro de un rango $[\alpha, \beta]$, generado a partir de una distribución uniforme $\mathcal{U}[\alpha, \beta]$. Para este tipo de ataque se han considerado dos variantes: RSA 0.25 1.1 y RSA 0.5 3 que representan diferentes grados de manipulación.

La primera de las variantes tiene el mismo objetivo que el resto de ataques, es decir, reportar datos de consumo por debajo de los reales. Sin embargo, el de la segunda variante, es declarar más de lo realmente consumido. La motivación de estos puede parecer contradictoria, considerando que el objetivo del atacante siempre es pagar menos de lo consumido, pues bien, podría haber dos razones por las que realizar este tipo de ataques:

1. El atacante puede ser una empresa competidora de la empresa víctima que quiere causar bajas de clientes de estas dañando su reputación *(i.e. loss of business)*

2. El atacante puede ser un cliente que lanza un ataque de camuflaje donde manipula sus propios datos de consumo (declarar menos de lo consumido) y el del vecino (declara más de lo consumido) de forma que la suma total de consumo de los dos corresponde al consumo real conjunto. Esto sirve para evitar los detectores que actúan a nivel de red local de vecinos *(Neighborhood Area Network)*.

2.2.4. Avg

El ataque *Average* (Avg) toma la media del consumo de cada semana y esta se utiliza para remplazar todos los registros de esa misma semana por el resultado de multiplicar el promedio por un número aleatorio dentro de [0.5, 1.5]. Este intervalo se utiliza con el objetivo de introducir cierto grado de variabilidad en los valores modificados de esa semana para intentar no ser detectado.

2.2.5. Swap

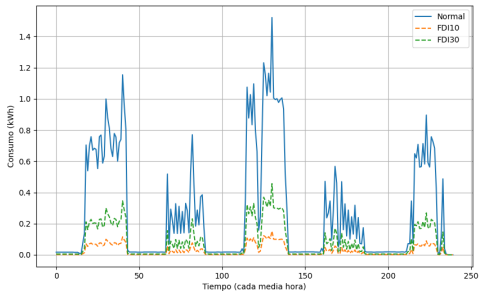
El escenario *Swap* se basa en que el precio del *kWh* varía según la hora del día. Para este ataque se van a distinguir dos franjas horarias: desde las 00:00 hasta las 9:00 (baja demanda) y desde las 9:00 hasta las 00:00 (alta demanda). El ataque consiste en cambiar los 18 consumos más altos del intervalo de alta demanda por todos los consumos del otro intervalo. De esta manera, el valor total del consumo diario no es alterado, por lo que hace que sea más complicado detectarlo.

Para concluir con esta subsección, en la Tabla 2.1 se recopilan, para cada escenario de consumo, el coste total de la factura [1] y la ganancia del atacante, o lo que es lo mismo, la pérdida de ganancia económica por parte de la empresa que gestiona la red. Para esta comparativa se han utilizado los datos del contador con identificador #1021 del *dataset* de la electricidad a lo largo de las primeras 60 semanas.

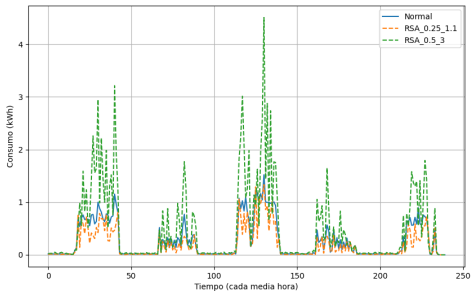
Escenario	**Coste total (€)**	**Ganancia del atacante**
Normal	1081.79	0 %
FDI_10	108.18	90 %
FDI_30	324.54	70 %
RSA_0.25_1.1	730.37	32 %
RSA_0.5_3	1897.45	75 %
Avg	1060.01	2 %
Swap	982.43	9 %

Tabla 2.1: Escenarios de consumo, coste total y ganancia del atacante

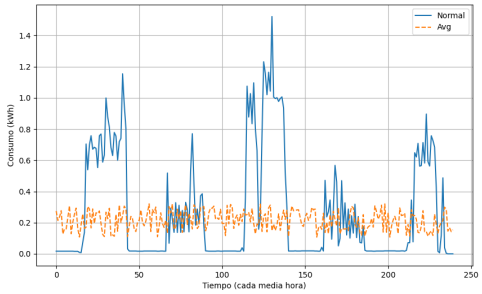
Por último, de forma similar, en la Figura 2.1 se observa gráficamente la diferencia entre los valores de consumo registrados de un escenario a otro. Estos datos de consumo se han obtenido a partir de los cinco primeros días del identificador #1021 del conjunto de datos de la electricidad para cada uno de los escenarios de consumo.



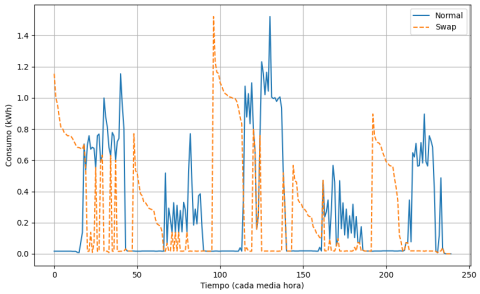
(a) Normal y FDI



(b) Normal y RSA



(c) Normal y Avg



(d) Normal y Swap

Figura 2.1: Diferencia del consumo en el escenario normal y en los escenarios de ataque

[1] Los datos utilizados para los cálculos provienen de `https://selectra.es/energia/info/que-es/precio-kwh`

2.3. Escenarios de generación

En este apartado se describen los diferentes escenarios de generación que nacen alrededor de la energía solar. Esto se debe a que las redes inteligentes permiten no solo consumir, sino también aportar el excedente de energía a la red (cantidad en kWh producida por los paneles solares y que no se gasta). Por tanto, tal como se ha hecho antes, distinguimos entre dos escenarios, el escenario normal, que tiene los valores registrados inalterados, y los escenarios de ataques de generación, donde se modifican los registros para reportar más energía de la realmente producida con el objetivo de obtener un beneficio económico.

2.3.1. Normal

El escenario normal no cambia respecto a los escenarios de consumo, este representa registros de consumo que no han sufrido alteraciones o manipulaciones, pero en este caso de generación de energía solar.

2.3.2. Random Scale Attack

Se vuelve a utilizar los ataques de tipo *Random Scale Attack* (RSA), concretamente RSA 0.5 3, ya que utilizando el rango [0.5,3] se declara más kilovatio hora de lo que realmente se genera. En el caso de ataques de consumo se utilizaba también este ataque, pero con la intención de desestabilizar la red, en este caso se utiliza con la única intención de obtener beneficio económico gracias a este reporte.

2.3.3. Rating

El escenario de ataque *Rating* [7] consiste en alterar los valores registrados desde las horas del amanecer hasta el atardecer, sustituyéndolos por el valor de la capacidad máxima del panel solar que está generando la energía. El resto de horas, dado que no hay luz solar, la generación se establece a cero, ya que nunca podría exceder dicho valor. Por lo tanto, la cantidad de energía robada es la diferencia entre la capacidad del panel y la generación real.

2.3.4. Percentile

El ataque *Percentile* [7], a diferencia del anterior, sí que tiene en cuenta las fluctuaciones diarias de la generación solar. Este escenario consiste en comparar cada registro con otros de la misma media hora en días anteriores. A partir de estas lecturas históricas, se obtiene el umbral percentil 99: el valor debajo del cual están el 99 % de

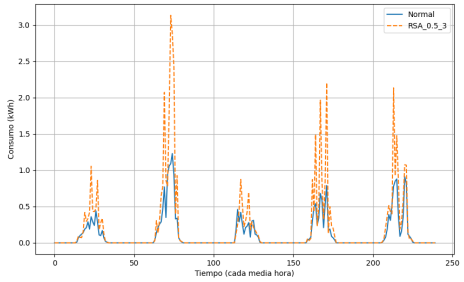
estas. El ataque consiste en sustituir la lectura real por este valor, en consecuencia, la cantidad robada es la diferencia entre el umbral y la generación real.

Finalmente, la Tabla 2.2 recopila, para cada escenario de generación, el excedente total [2], es decir, la cuantía de dinero que gana gracias a la generación de la energía que no consume. Para esta comparativa se han utilizado los datos del contador con identificador #1 del *dataset Solar Ausgrid* a lo largo de las primeras 50 semanas.

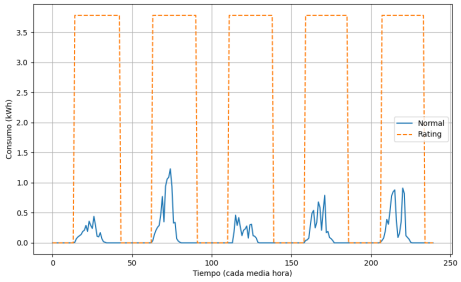
Escenario	**Excedente total (€)**	**Ganancia del atacante**
Normal	116.37	0 %
RSA_0.5_3	204.75	76 %
Rating	1282.49	1002 %
Percentile	414.44	256 %

Tabla 2.2: Escenarios de generación, excedente total y ganancia del atacante

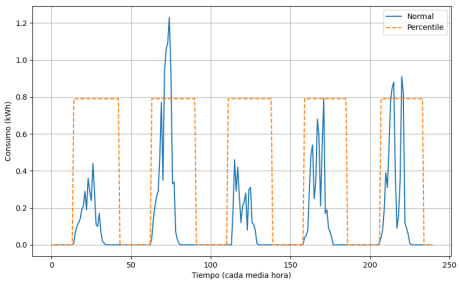
Por último, en la Figura 2.2 se observa gráficamente la diferencia entre los valores de generación registrados de un escenario a otro. Estos datos de generación se han obtenido a partir de los cinco primeros días del identificador #1 del *Ausgrid Solar Dataset* para cada uno de los escenarios de generación.



(a) Normal y RSA



(b) Normal y Rating



(c) Normal y Percentile

Figura 2.2: Diferencia de la generación en el escenario normal y en los de ataque

[2]Los datos utilizados para los cálculos provienen de `https://selectra.es/autoconsumo/info/tarifas/pvpc`

Capítulo 3

Detectores

En este capítulo se explican las técnicas y modelos utilizados para detectar los ataques explicados en el capítulo anterior. Se incluyen tanto detectores previamente evaluados, pero que son reevaluados (3.1), como nuevas incorporaciones. Entre estas últimas, se encuentran detectores ya implementados que no habían sido evaluados en el contexto de las *smart grids* (3.2), nuevas propuestas basadas en técnicas más avanzadas (3.3) y optimizaciones de ciertos modelos preexistentes (3.4).

Nombre	**Técnica**
MinAverage	Comparación del mínimo de los consumos medios en el pasado
ARIMA, ARIMAX	Modelos autorregresivos
KLD, JSD	Comparación de distribuciones de frecuencias relativas de consumo
Tegdet	Grafos de evolución temporal
NN, NN_v2, LSTM	Redes neuronales

Tabla 3.1: Detectores considerados en la experimentación

3.1. Detectores iniciales

En esta sección se describen los detectores implementados en [2].

3.1.1. MinAverage

El detector *MinAverage* [8] utiliza una serie de datos de consumo recopilados por un contador durante un período de entrenamiento de n semanas. La predicción de los detectores se basa en calcular el consumo promedio para cada semana de entrenamiento y determinar el valor m, que se refiere al valor mínimo de todos los promedios de las n semanas de entrenamiento. Cualquier semana de *testing* cuyo consumo promedio sea inferior a m será identificada como anómala.

3.1.2. ARIMA y ARIMAX

ARIMA (*Auto-Regressive Integrated Moving Average*) [7] es un modelo estadístico que, dado una serie temporal de datos, realiza predicciones de los valores futuros basándonos en los valores pasados. Se caracteriza por los siguientes parámetros que cuanto más precisos son, mejor se ajusta el modelo:

- p: orden de autorregresión
- q: orden de la media móvil
- d: grado de diferenciación de la serie temporal

Una vez obtenido el modelo *ARIMA*, este se utiliza para predecir el consumo cada media hora, con un intervalo de confianza del 95 %. Por tanto, un valor registrado en un momento t es considerado normal si y solo si se encuentra dentro del intervalo de confianza de la predicción realizada en t, de lo contrario es considerado anómalo.

El detector *ARIMAX* (*Auto-Regressive Integrated Moving Average with eXplanatory variable*) [2] es una variante de *ARIMA* que funciona exactamente igual, pero que incluye variables exógenas de Fourier para tener en cuenta la frecuencia diaria del consumo.

3.1.3. KLD y JSD

Los detectores *KLD* [7] y *JSD* [2] se basan en la comparación de distribuciones de frecuencia relativa del consumo registrado por un contador. Estos, al igual que *Min-Avg* trabajan a nivel de semana, "etiquetan una semana de consumo como anómala si su

distribución se desvía 'demasiado' de la distribución de frecuencia relativa de consumo histórica, es decir, de la distribución obtenida de los datos de entrenamiento" [9].

La diferencia entre *KLD* y *JSD* radica únicamente en la función utilizada para cuantificar la divergencia de las dos distribuciones: *Kullback-Leibler Divergence* y *Jensen-Shannon Divergence*, respectivamente.

3.1.4. NN

El detector *NN* (*Neural Network*) se distingue del resto de los detectores por seguir un enfoque de aprendizaje supervisado. Gracias a este, no solo es capaz de identificar si un consumo es anómalo o no, sino que además determina a qué tipo de escenario corresponde cada muestra. Por esto, el modelo necesita ser entrenado para todos los escenarios considerados, tanto el escenario normal como los diferentes escenarios de ataques descritos en las Secciones 2.2 y 2.3.

Para el entrenamiento de la red neuronal, se reducen las 48 muestras diarias (una cada media hora) a una única muestra que representa el día completo. Aunque se podría haber optado por representar ese día mediante el valor medio del consumo registrado a lo largo del mismo, se optó por incluir más información al modelo con la intención de mejorar su capacidad de clasificación. Por ello, se consideraron los siguientes calificadores estadísticos:

- Media
- Moda
- Desviación típica
- Varianza
- Valor máximo - valor mínimo
- Coeficiente de variación
- *Skewness*
- Q1, Q2, Q3
- Rango intercuartil
- Última muestra - primera muestra

El modelo de predicción consiste en una red neuronal compuesta por dos capas. La primera capa con 50 neuronas y la segunda con N neuronas, donde N es el número de escenarios: 7 para los casos de consumo y 4 para los de generación. La capa de salida tiene una función de activación *softmax* que da como resultado un vector cuyas componentes representan cada una de las clases y cuyos valores son las probabilidades de pertenencia (del día evaluado) a cada una de estas. El tipo de ataque se determina a partir de la clase cuya probabilidad sea más alta, es decir, la componente del vector cuyo valor es más alto.

3.2. Tegdet

Los detectores *Tegdet* propuestos en [5], son herramientas diseñadas con el fin de analizar series temporales y encontrar anomalías en las mismas. Se basan en *Time Evolving Graphs* (*TEGs*), es decir, grafos que capturan la evolución de una serie temporal.

Cada grafo representa el uso del contador durante una semana. Los nodos del grafo representan la cantidad consumida, mientras que las aristas representan cambios en el consumo. Cada nodo y arista del grafo tienen pesos basados en su frecuencia de aparición. Para crear estos grafos es necesario hacer un paso previo similar al de los detectores *KLD* y *JSD* para convertir los datos de consumo de *kWh* a niveles (*p.ej.:* nivel 0, nivel 1, etc.).

El criterio de detección de anomalías consiste en coger cada subsecuencia, es decir, cada grafo que representa una semana de *testing*, y compararla con el grafo de todo el periodo de entrenamiento. Si la diferencia de ambos es mayor al umbral percentil $(100-\alpha)$ entonces esa semana se identifican como anómala, indicando posibles manipulaciones.

Tal y como se documenta en [5] los atributos clave para crear un detector son:

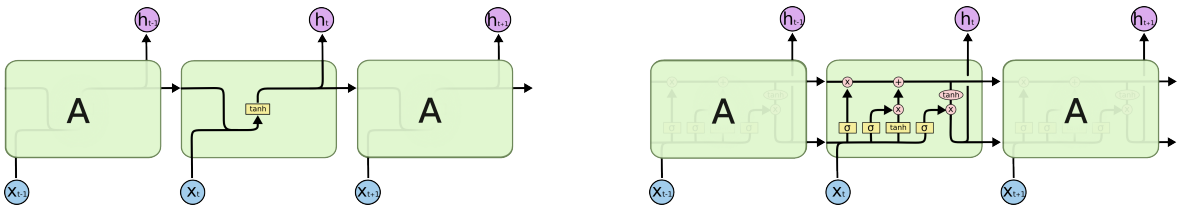
- ***metric***: nombre de la métrica de disimilitud para comparar grafos
- ***n_bins***: representa el número de niveles en los que se mapean las observaciones de la serie. Su valor predeterminado es 30
- ***n_obs_per_period***: indica el número de observaciones en cada período de la serie. Su valor predeterminado es 336
- ***alpha***: umbral para la detección de anomalías. Una época se considera anómala si sus observaciones difieren significativamente del modelo de referencia. Su valor predeterminado es 5

Para cada uno de los detectores estudiados, se utilizan los valores predeterminados para sus parámetros: *n_bins* (30), *n_obs_per_period* (336) y *alpha* (5). En cuanto a la métrica de disimilitud, se consideran todas las opciones disponibles, que da como resultado un total de 28 detectores correspondientes a las diferentes métricas usadas.

3.3. LSTM (Long Short-Term Memory)

Las redes neuronales *LSTM (Long Short-Term Memory)* [6], son un tipo de red neuronal recurrente *(RNN)* que se caracteriza por la captura de dependencias a largo plazo. Están diseñadas específicamente para aprender y recordar información a lo largo de secuencias de datos, lo que las hace un modelo muy interesante para la predicción de datos de series temporales.

En términos de la estructura, las *LSTM* son bastante similares a las *RNN*, ya que ambas presentan una configuración de cadena con módulos repetitivos de redes neuronales. Sin embargo, este módulo en el caso de las *LSTM* consta de cuatro capas que interactúan entre sí, mientras que en las *RNN* presentan una única capa, tal y como se puede observar en la Figura 3.1.



(a) (b)

Figura 3.1: Estructura interna de RNN (a) y LSTM (b). Fuente: [10]

La *LSTM* posee un elemento clave representado por una línea horizontal en la parte superior del diagrama de red llamado estado de la celda. Este estado de la celda funciona como una especie de "memoria" para la red, lo que le permite ser capaz de retener información relevante y descartar aquella información irrelevante mediante estructuras llamadas puertas o *gates*.

Las puertas sirven para permitir opcionalmente que la información pase a través de ellas. Estas están compuestas de una capa de red neuronal sigmoide y una operación de multiplicación punto a punto. Las *LSTM* cuentan con tres puertas sigmoidales: puerta de olvido o *forget gate* (f_t), puerta de entrada o *input gate* (i_t) y puerta de salida u *output gate* (o_t).

- **Puerta de olvido (f_t)**: responsable de decidir qué información del estado de la celda debe conservarse o descartarse. Utiliza una función sigmoide a la que le pasa la información del estado oculto anterior y de la entrada actual. Esta función devuelve un valor, los valores cercanos a cero indican que la información debe olvidarse y los valores cercanos a uno significan que la información debe conservarse

- **Puerta de entrada (i_t)**: determina qué información se almacenará en el estado de la celda. Está formada por dos partes: una capa sigmoide que decide qué valores deben actualizarse y una capa *tanh* que genera un vector de valores candidatos que podrían agregarse al estado de la celda

- **Puerta de salida (o_t)**: determina qué información del estado de la celda actual debe enviarse como salida. Esto se consigue multiplicación entre la puerta de salida y la función *tanh* del estado de celda actual

Las distintas funciones de transición de *LSTM* se definen en las siguientes ecuaciones [10]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.3}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.5}$$

$$h_t = o_t * \tanh(C_t) \tag{3.6}$$

Donde σ es la función *sigmoide*, *tanh* es la función tangente hiperbólica, W_f, W_C, W_i y W_o son matrices de pesos, b_f, b_i, b_C y b_o son términos de sesgo, y x_t es la entrada en el tiempo t.

3.3.1. Ventana deslizante

En este modelo se emplea una técnica muy común en las redes neuronales *LSTM* conocida como "ventana deslizante". Esta técnica consiste en considerar subconjuntos de datos en ventanas de tiempo de tamaño fijo que se solapan. Es decir, para una ventana con tamaño N se toman N muestras de la serie temporal para intentar predecir el valor de la muestra en $N + 1$. Tras esto, la ventana se desplaza una posición de manera que ahora busca predecir el valor de la muestra $N + 2$. Este desplazamiento de la ventana a lo largo de la serie temporal se repite hasta que esta haya sido recorrida por completo. El funcionamiento se ilustra con mayor claridad en la Figura 3.2.

Esta ventana permite que el modelo *LSTM* aprenda las dependencias temporales de los datos y, en consecuencia, mejore su capacidad para hacer predicciones. Un aspecto importante de esta ventana es la selección de su tamaño óptimo, ya que de este dependerá la capacidad del modelo para capturar patrones temporales en los datos.

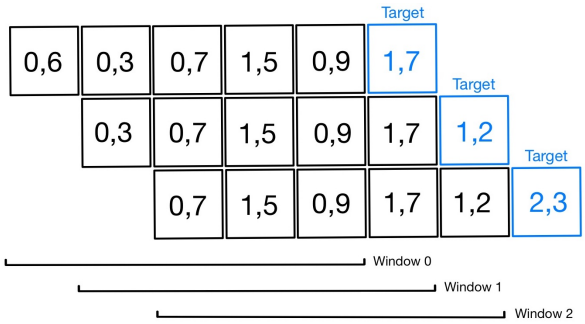


Figura 3.2: Funcionamiento de la ventana deslizante

Sin embargo, el tamaño de la ventana no es el único factor que influye en la precisión del modelo. Además de utilizar el valor del consumo registrado en cada media hora, es posible mejorar la precisión de las predicciones mediante la inclusión de calificadores estadísticos que doten de más información al modelo. Estos calificadores fueron elegidos tras horas de experimentación en las que se probaron distintas versiones del modelo utilizando diferentes *features* tal y como se ilustra en el Anexo F.4.2. La combinación que mejores resultados obtuvo se resumen a continuación:

- Media de los valores registrados ese día
- Moda de los valores registrados ese día
- Valor máximo del día
- Valor mínimo del día
- Desviación típica del día
- Coeficiente de variación del día
- Valor del día anterior en la misma media hora
- Valor del día anterior en la siguiente media hora
- Valor previo registrado
- Valor actual - Valor previo
- Media de los valores registrados hasta el momento en la misma media hora que la actual
- Valor máximo de los valores registrados hasta el momento en la misma media hora que la actual
- Valor mínimo de los valores registrados hasta el momento en la misma media hora que la actual

Volviendo al tema de la selección del tamaño de la ventana, *a priori*, un tamaño de ventana más pequeño proporciona una visión más detallada de los cambios inesperados, permitiendo detectar mejor los picos de consumo. Sin embargo, es posible que no capture algunos patrones que se repiten con el tiempo. Por otro lado, una ventana más grande proporciona una perspectiva más general, pero puede sacrificar algo de precisión en los cambios de comportamiento a corto plazo. En definitiva, el tamaño más adecuado para dicha ventana dependerá de los datos con los que se esté trabajando.

Por ello, con la intención de comprobar que tamaño de ventana se adapta mejor a nuestros datos, se realizaron pruebas con diferentes tamaños de ventana (2, 3, 5, 7, 10, 14 y 30). Estos tamaños son la cantidad de muestras que se tiene como referencia para

intentar predecir el valor registrado en la muestra actual. En este caso, se trabaja con lecturas cada 30 minutos, por tanto, si el tamaño de la ventana es igual a 5, esto quiere decir que a partir de las muestras registradas en las cuatro medias horas previas a la actual, junto a la lectura actual, se intenta adivinar el valor registrado en la siguiente media hora: $((N-4),(N-4),\ldots,(N) \rightarrow (N+1))$.

Para esta experimentación se seleccionaron aleatoriamente 10 contadores de cada uno de los tres conjuntos de datos disponibles, es decir, 30 contadores en total. Para cada contador se realizaron pruebas de los siete escenarios de consumo que se definen en la Sección 2.2, además, para los contadores del conjunto de datos Solar se probaron también los cuatro escenarios de generación definidos en la Sección 2.3. A partir de estos datos se probó el mismo modelo *LSTM* únicamente cambiando el tamaño.

De acuerdo con los resultados resumidos en la Tabla 3.2 (ver las tablas completas en F.4.2), el tamaño escogido para la versión definitiva del modelo es la ventana de 5 días, cuya *accuracy (n.º de aciertos / n.º de observaciones)* es algo más alta que en el resto de ventanas. A pesar de esta elección, es importante señalar que como se puede observar, en nuestro caso concreto, el tamaño de la ventana parece no implicar una diferencia significativa en el modelo.

	Observaciones	**Aciertos**	**Accuracy**
LSTM_2	126604300	103435713	0.817
LSTM_3	126604050	103815321	0.820
LSTM_5	126603550	104194721	0.823
LSTM_7	126603050	103308088	0.816
LSTM_10	126602300	103307476	0.816
LSTM_14	126601300	103306660	0.816
LSTM_30	126597300	103303396	0.816

Tabla 3.2: Resultados del modelo *LSTM* dependiendo del tamaño de la ventana

3.3.2. Detección de Anomalías basadas en la predicción

Similarmente, a como se propone en [11] nuestro modelo se divide en dos fases: una de predicción y otra de detección de anomalías basándose en el valor de consumo predicho. En la sección anterior, se describió la primera fase en la que, mediante una ventana de valores de consumo, se trataba de predecir el siguiente valor de la serie temporal. Para ilustrar este proceso, en la Figura 3.3 se muestra una comparación entre los valores de consumo reales, y los valores de consumo predichos por el modelo. Estos resultados recogen las cien primeras muestras, una cada media hora, de los datos de *testing* de un escenario normal, registradas por el contador con identificador #1014 del *dataset* de electricidad.

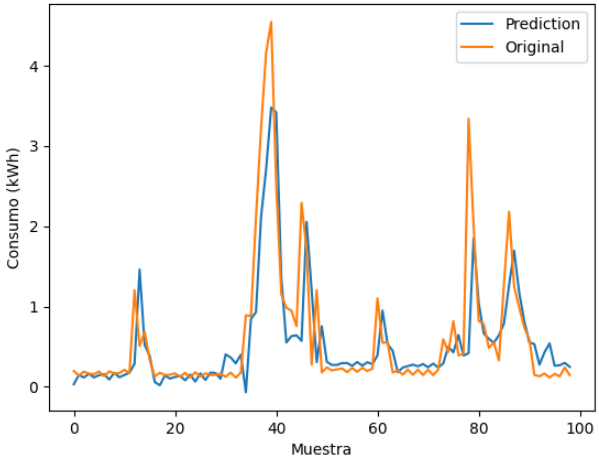


Figura 3.3: Gráfica de los valores originales y los valores predichos por el modelo

Una vez que se han obtenido todas las predicciones del modelo, se comparan estos valores con el valor real y entonces se pasa a la fase de detección de anomalías. El criterio de detección utilizado se basa en un umbral predefinido (*threshold*), en concreto se calcula el valor absoluto de la resta entre el valor predicho y el original, si este resultado es menor que el valor del *threshold*, entonces el valor se encuentra dentro de la zona de tolerancia y, por tanto, se clasifica como un valor normal. En caso contrario, es decir, de superar el umbral, esta muestra se clasifica como anómala. Tal y como se muestra en la Ecuación 3.7 y de manera más visual en la Figura 3.4.

$$\text{Anomalía} = \begin{cases} \text{No} & \text{si } |y_{\text{pred}} - y_{\text{real}}| \leq \textit{threshold} \\ \text{Sí} & \text{si } |y_{\text{pred}} - y_{\text{real}}| > \textit{threshold} \end{cases} \tag{3.7}$$

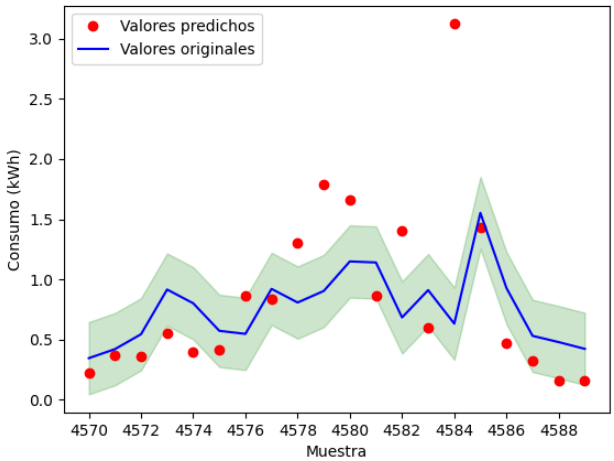


Figura 3.4: Gráfica que representa como funciona la detección de anomalías

Es importante señalar que este modelo ha sido entrenado y probado utilizando datos del mismo escenario. Por tanto, al entrenar el modelo con datos de un escenario normal, las anomalías que se detecten implican que el modelo considera que ese valor es fruto de un ataque (falsos positivos). Por otro lado, en el caso de los escenarios de ataque, las anomalías detectadas indican que los valores no se reconocen como parte de ese escenario de ataque específico (falsos negativos).

La repartición de los datos en conjuntos de *training* y *testing*, así como otras características importantes de los distintos *datasets* con los que se han trabajado, se explican en el Anexo B.

3.3.3. Determinación del threshold

El proceso para determinar el valor numérico del *threshold* es un paso crucial en la definición del modelo, ya que, valores muy altos o muy bajos de dicho umbral afectan directamente a la calidad del mismo (ver Anexo A.3 para más detalles).

Por tanto, con la intención de determinar un valor adecuado, se llevó a cabo una experimentación que consistió en seleccionar 10 contadores del conjunto de datos *ISSDA CER* Electricidad y a partir de esta lista de contadores, se entrenó individualmente al modelo con los ficheros de entrenamiento correspondientes al escenario normal de cada contador. Una vez completado el entrenamiento, los ficheros que contienen los datos correspondientes a la fase de *testing* se manipularon de tal manera que se inyectó ataques de tipo *FDI10* y *FDI30* en un 10 % aleatorio de las muestras.

Es decir, el modelo había sido entrenado para detectar escenarios sin ataques, sin embargo, es evaluado con los datos que, *a priori*, no tienen ataques pero que en realidad algunas muestras sí. De esta manera, se comprueba cuantos de estos ataques aislados es capaz de detectar el modelo dependiendo del valor establecido para el umbral.

El objetivo principal era encontrar un valor que equilibrase la precisión del modelo y los falsos positivos del mismo. Para ello, se probaron distintos valores y se observó, tal y como se representa en la Figura 3.5, como responde el modelo en términos de: tasa de falsos negativos, *accuracy* ante inyección de ataques y *accuracy* sin inyección de ataques. Tras analizar los resultados se decide seleccionar 0.3 como valor final, este valor mantiene cierto equilibrio entre el porcentaje de falsos negativos ($\approx 40\,\%$) y la precisión del modelo ($\approx 80\,\%$).

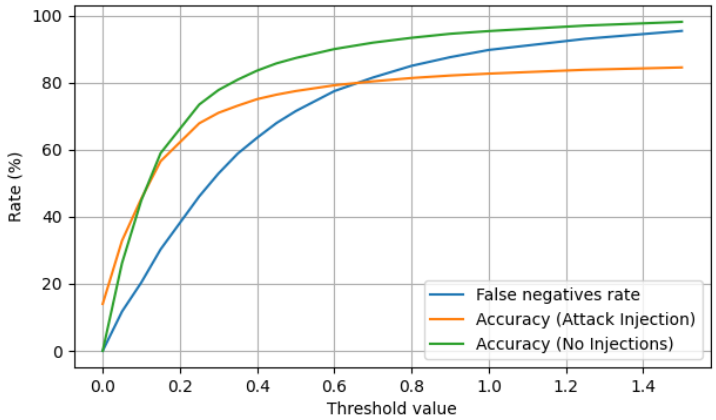


Figura 3.5: Variación de las distintas métricas dependiendo del valor del *threshold*

Las distintas métricas empleadas para esta experimentación se detallan en la Tabla 3.3, pero primero hay que definir tp, fp, tn y fn:

- **Verdaderos negativos** (tn): número de observaciones no falsificadas y correctamente clasificadas como tal
- **Falsos positivos** (fp): número de observaciones no falsificadas y erróneamente clasificadas como anómalas
- **Verdaderos positivos** (tp): número de observaciones falsificadas y correctamente clasificadas como anómalas
- **Falsos negativos** (fn): número de observaciones falsificadas y erróneamente clasificadas como valores normales

A partir de estos valores se calculan las distintas métricas que se identifican en la Figura 3.5: *accuracy* (*attack injection*), *accuracy* (*no injections*) y *false negative rate*. La primera de estas es la precisión del modelo al ser entrenado con escenarios normales pero evaluado con ataques inyectados en las muestras normales. La segunda representa la precisión cuando es entrenado y evaluado con escenarios sin ningún tipo de inyección de ataques ni manipulación. La tercera y última se refiere al número de falsos negativos detectados respecto al total de ataques inyectados, es decir, un 40 % quiere decir que si se han inyectado 100 ataques se están detectando 60.

accuracy (attack injection)	$(tp+tn)/(tp+fn+fp+tn)$
accuracy (no injections)	$(tn)/(fp+tn)$
false negative rate	$fn/(tp+fn)$

Tabla 3.3: Métricas empleadas

3.3.4. Optimización de hiperparámetros

Los hiperparámetros son valores específicos que se establecen antes de entrenar un modelo y que influyen en la forma en la que este se entrena. El ajuste u optimización de estos consiste en encontrar la combinación de valores óptima para un algoritmo de aprendizaje, en nuestro caso, el modelo *LSTM*. "Esta combinación de hiperparámetros maximiza el rendimiento del modelo, minimizando una función de pérdida predefinida para producir mejores resultados" [12]. Los hiperparámetros que se han probado son los siguientes:

- **Número de neuronas por capa**: tener una mayor cantidad de neuronas en cada capa no indica necesariamente que nuestro modelo sea mejor. De hecho, es importante conseguir un número de neuronas lo suficientemente pequeño como para evitar que el modelo memorice los datos de entrenamiento y, en consecuencia, no sea capaz de generalizar bien con los datos de prueba (*overfitting*). Y, al mismo tiempo, el número de neuronas ha de ser lo suficientemente grande como para que el modelo sea capaz de aprender patrones a partir de los datos de entrenamiento, evitando así el *underfitting*.

- **Número de capas ocultas**: igual que sucede con el número de neuronas, mayor cantidad de capas del modelo no siempre se traduce en un mejor rendimiento. Se trata de encontrar un equilibrio entre la simplicidad del mismo y la capacidad de predecir correctamente.

- **Tamaño del lote**: el tamaño de lote o *batch size* es un hiperparámetro que "define el número de muestras que se deben procesar antes de actualizar los parámetros internos del modelo" [13]. Este puede ser uno de los pasos claves para asegurar el rendimiento óptimo del modelo.

Para poder llevar a cabo la optimización de estos valores, se ha utilizado el módulo *Keras Tuner* [14] que "es un marco de optimización de hiperparámetros escalable y fácil de usar que permite utilizar diferentes algoritmos de búsqueda tales como: optimización bayesiana, hiperbanda y búsqueda aleatoria" [14]. El algoritmo escogido en nuestro caso ha sido el de hiperbanda [15]. El proceso de búsqueda de la mejor combinación de hiperparámetros para el modelo se resume en la Figura 3.6.

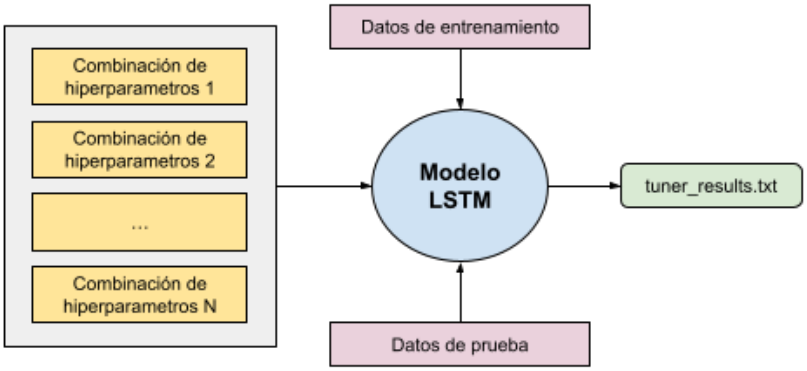


Figura 3.6: Diagrama del funcionamiento de la optimización de hiperparámetros

Tras analizar los resultados obtenidos en el fichero de salida (mostrado en el Anexo F.4.2), se han definido los valores finales para los hiperparámetros. Estos resultados provienen de la búsqueda de hiperparámetros llevada a cabo en 10 contadores seleccionados aleatoriamente para cada uno de los conjuntos de datos: electricidad, gas y de solar. Los 10 contadores del solar son evaluados dos veces, una para los escenarios de generación y otra para los de consumo, es decir, se recogen en total 40 resultados distintos.

El número de capas ocultas se ha determinado identificando el valor que más se ha repetido en los resultados obtenidos, mientras que el número de neuronas por capa *LSTM* y el *batch size* se han definido calculando la media de todos los resultados.

Hiperparámetro	**Valor**
Capas ocultas	3
Neuronas *LSTM*	280
Tamaño del lote	66

Tabla 3.4: Valores óptimos para los hiperparámetros

En resumen, la arquitectura final del modelo consiste en: una capa de entrada seguida de tres capas ocultas y una capa de salida. Las capas ocultas se componen de tres capas *LSTM* con 280 neuronas y con función de activación *relu*, estas capas estas separadas entre sí por capas de *Dropout* con un *dropout rate* de 0.2 para prevenir el *overfitting*. Por último, la capa de salida es una capa *Dense* con una única 1 neurona que tiene como función sacar el valor predicho.

3.4. Red Neuronal optimizada

La optimización de la red neuronal *(NN)*, tal y como se comenta en la Sección 1.2, fue uno de los objetivos principales de la realización de este trabajo.

El modelo original, a pesar de ser el modelo más prometedor de todos los evaluados inicialmente gracias a su excelente capacidad de detección de todos los tipos de ataques, este presentaba una limitación que impedía explotar todo su potencial: el *overfitting*. Este fenómeno ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento y pierde la capacidad de generalizar correctamente en datos no vistos. Como consecuencia, el modelo presentaba un alto número de falsos positivos, ya que no era capaz de distinguir entre ciertos tipos de ataque y el escenario normal. Para abordar este problema, se implementaron dos estrategias de optimización: la técnica de la ventana deslizante y la optimización de hiperparámetros.

3.4.1. Ventana deslizante

La técnica de ventana deslizante permite que la red neuronal no se base únicamente en la información inherente a una muestra específica, representada por sus calificadores estadísticos, sino que también incorpora información de las muestras de días anteriores.

Concretamente, para clasificar un día X, se conoce la información de los $N-1$ días anteriores y del mismo día, siendo N el tamaño de la ventana. Con ese lote de datos donde se guarda la información de los *features* de los $N-1$ días anteriores y del día actual, se intenta clasificar el día actual en uno de los escenarios para los que ha sido entrenado.

Un paso muy importante es el de elegir el tamaño adecuado para la ventana, por ello, de la misma manera que en la Sección 3.3.1, se realizaron pruebas con diferentes tamaños de ventana con la intención de ver como este afecta en la precisión del modelo, específicamente se probaron ventanas de 2, 3, 5, 7, 10, 14 y 30 días.

La experimentación ha sido la misma que para *LSTM*, la única diferencia es que en el anterior modelo las predicciones se hacían cada media hora, mientras que ahora se realizan por día, es decir, el número de observaciones es menor. Concretamente, antes, utilizando los datos de un contador del conjunto de datos de la electricidad, se trabaja con 5040 muestras para cada escenario (15 semanas $\times$ 7 días $\times$ 48 medias horas). Ahora, el número de muestras se ve reducido a 105 (15 semanas $\times$ 7 días). A continuación, se muestra un resumen de los resultados obtenidos, las tablas completas se encuentran en el Anexo F.4.1.

	Observaciones	Aciertos	Accuracy
NN_v2_2	54450	48950	0.899
NN_v2_3	54200	49267	0.909
NN_v2_5	53700	49028	0.913
NN_v2_7	53200	48784	0.917
NN_v2_10	52450	48201	0.919
NN_v2_14	51450	47488	0.923
NN_v2_30	47450	44033	0.928

Tabla 3.5: Resultados del modelo *NN_v2* dependiendo del tamaño de la ventana

Como se muestra en la Tabla 3.5, los resultados indican una mejora en la *accuracy (n.º de aciertos / n.º observaciones)* a medida que el tamaño de la ventana aumenta. Sin embargo, hay una consideración importante a tener en cuenta: cuanto mayor es la ventana, menor es el número total de predicciones del modelo.

Por ejemplo, si se elige una ventana de 30 días, las primeras 30 muestras no pueden ser clasificadas ya que no tienen información de los días anteriores. Esto significa que la primera muestra en ser clasificada es la del día 31. En el contexto del conjunto de datos de la electricidad, que cuenta con 105 días de *testing*, si se eliminan 30 predicciones, implica una reducción del 29 % de las predicciones totales.

Por lo tanto, aunque una ventana más grande puede mejorar la precisión, es muy importante encontrar un punto de equilibrio entre esto y la necesidad de mantener un número significativo de predicciones. Teniendo esto en cuenta, se elige la ventana de 7 días como la definitiva para la evaluación de este modelo.

3.4.2. Optimización de hiperparámetros

Para este proceso se han seguido los mismos pasos que en la Sección 3.3.4 del modelo *LSTM* pero, en este caso, adaptado a la red neuronal. Se han probado los mismos tipos de hiperparámetros: número de capas ocultas, número de neuronas y tamaño de lote. Y se ha realizado la misma experimentación: probar todas las combinaciones de valores posibles para los hiperparámetros en 30 contadores de los distintos *datasets* y extraer la combinación que mejores resultados obtiene para cada contador.

Al igual que anteriormente, el número de capas ocultas se determina identificando el valor que más se ha repetido en los resultados obtenidos, mientras que el número de neuronas por capa y el *batch size* se define calculando la media de todos los resultados.

Hiperparámetro	**Valor**
Capas ocultas	2
Neuronas *Dense*	320
Tamaño del lote	50

Tabla 3.6: Valores óptimos para los hiperparámetros

Finalmente, la arquitectura del modelo consiste en: una capa de entrada seguida de dos capas ocultas y una capa de salida. Las capas ocultas se componen de dos capas *Dense* con 320 neuronas y con función de activación *relu*, estas capas estas separadas entre sí por capas de *Dropout* con un *dropout rate* de 0.2 para prevenir el *overfitting*. Por último, la capa de salida es una capa *Dense* con N neuronas, siendo N el número de escenarios para los que la red neuronal es entrenada (7 para los de consumo y 4 para los de generación), la capa tiene una función de activación *softmax* cuya función es dar una probabilidad de pertenencia de la muestra que se está evaluando a cada una de las clases o escenarios.

Capítulo 4

Experimentación

La experimentación llevada a cabo en este trabajo consta de varias fases. Primeramente, la repetición de los experimentos realizados en [2] utilizando los mismos modelos y escenarios de ataque (Sección 4.1). A continuación, la validación de los resultados previos mediante la experimentación de los modelos con un nuevo conjunto de datos que introduce el concepto de los ataques de generación (Sección 4.2). Por último, evaluar nuevos detectores: unos ya implementados pero no evaluados en el contexto de las redes inteligentes (Sección 4.3) y otros completamente nuevos (Sección 4.4). Estos han sido evaluados con los conjuntos de datos anteriores (electricidad y gas) para los escenarios de consumo y con el nuevo *dataset* (solar) para los escenarios de consumo y de generación.

En el Anexo B se puede encontrar más información acerca de los conjuntos de datos que se han utilizado. Concretamente, acerca del número de contadores considerados y de la división de los datos en entrenamiento y prueba, es decir, del número total de semanas, cuáles han sido utilizadas para entrenar los modelos y cuáles para evaluarlos. Además, en el Anexo E, se detallan cada una de las métricas consideradas para evaluar la eficiencia de los detectores.

Antes de presentar los resultados de los experimentos, primero se definen en la Tabla 4.1 los diferentes planes de la experimentación que se han enumerado con anterioridad. De este modo, se pueden analizar visualmente las distintas combinaciones de *datasets*, escenarios, número de contadores y número de detectores empleados en cada una de las fases para tener una idea de la complejidad de las mismas.

Experimento	*Dataset*	Escenarios	Contadores	Detectores
4.1	electricidad	consumo	1000	6
	gas	consumo	1000	6
4.2	electricidad	consumo	1000	6
	gas	consumo	1000	6
	solar	consumo	300	6
	solar	generación	300	6
4.3	electricidad	consumo	1000	28
	gas	consumo	1000	28
	solar	consumo	300	28
	solar	generación	300	28
4.4	electricidad	consumo	1000	2
	gas	consumo	1000	2
	solar	consumo	300	2
	solar	generación	300	2

Tabla 4.1: Fases de la experimentación y sus características

4.1. Reproducción de los resultados previos

En esta sección se resume la primera parte de la experimentación que consistió en repetir los experimentos de [2] utilizando los mismos conjuntos de datos (electricidad y gas) y los mismos modelos de detección de ataques (*Min-Avg, ARIMA, ARIMAX, KLD, JSD, NN*).

Las tablas completas de los resultados obtenidos se pueden encontrar en el Anexo F.1, a continuación, se muestran los resultados resumidos de la tasa de verdaderos positivos (*tpr*), la tasa de verdaderos negativos (*tnr*) y la *balanced accuracy* (*bacc*). Estos resultados han sido calculados como valores medios sobre dos conjuntos de datos de consumo (electricidad y gas) y no dependen de los escenarios de ataque específicos.

Modelo	*tnr*	*tpr*	*bacc*
Min-Avg	0.984	0.201	0.592
ARIMA	0.934	0.041	0.487
ARIMAX	0.932	0.059	0.495
KLD	0.899	0.519	0.709
JSD	0.919	0.499	0.709
NN	0.125	0.936	0.530

(a) Resultados iniciales

Modelo	*tnr*	*tpr*	*bacc*
Min-Avg	0.988	0.197	0.592
ARIMA	0.934	0.042	0.488
ARIMAX	0.931	0.059	0.495
KLD	0.903	0.523	0.713
JSD	0.922	0.501	0.711
NN	0.714	0.948	0.831

(b) Resultados de la repetición

Tabla 4.2: Resultados de los detectores iniciales considerando electricidad y gas

4.2. Validación de los resultados previos

Una vez confirmados los resultados anteriores, se procedió a validar los modelos utilizando un nuevo conjunto de datos: *Ausgrid Solar Dataset* [4]. Con esta experimentación se pudo comprobar la eficacia de los detectores en un contexto diferente de las redes inteligentes. De la misma forma que antes, las tablas completas de los resultados se pueden encontrar en el Anexo F.1.3.

Modelo	*tnr*	*tpr*	*bacc*
Min-Avg	0.940	0.378	0.659
ARIMA	0.940	0.043	0.491
ARIMAX	0.934	0.071	0.503
KLD	0.858	0.626	0.742
JSD	0.873	0.591	0.732
NN	0.780	0.975	0.878

(a) Escenarios de consumo

Modelo	*tnr*	*tpr*	*bacc*
Min-Avg	0.971	0.008	0.490
ARIMA	0.907	0.344	0.626
ARIMAX	0.907	0.339	0.623
KLD	0.914	0.730	0.822
JSD	0.907	0.686	0.797
NN	0.897	0.872	0.884

(b) Escenarios de generación

Tabla 4.3: Resultados de los detectores iniciales considerando el *dataset* solar

4.3. Evaluación de los detectores teg

Para ampliar la experimentación se introdujo un nuevo conjunto de detectores, los detectores "tegdet" [5], cuyo funcionamiento básico se detalla en la Sección 3.2 y cuyas tablas de resultados completas se pueden encontrar en el Anexo F.2.

Modelo	*tnr*	*tpr*	*bacc*
Hamming	0.926	0.537	0.731
Canberra	0.858	0.736	0.797
Lorentzian	0.936	0.010	0.473
KL	0.356	0.793	0.574
JS	0.878	0.607	0.742
Clark	0.885	0.662	0.773

(a) Escenarios de consumo

Modelo	*tnr*	*tpr*	*bacc*
Hamming	0.884	0.956	0.920
Canberra	0.857	0.997	0.927
Lorentzian	0.942	0.000	0.471
KL	0.228	0.994	0.611
JS	0.850	0.995	0.922
Clark	0.863	0.992	0.927

(b) Escenarios de generación

Tabla 4.4: Resultados de los detectores *teg* considerando el *dataset* solar

4.4. Evaluación de la red neuronal optimizada y *LSTM*

En esta sección se resumen los resultados obtenidos por los detectores: NN_v2 y *LSTM*. Las tablas completas se pueden encontrar en los Anexos A.7 y A.6.

Modelo	*tnr*	*tpr*	*bacc*
NN_v2	0.765	0.926	0.845
LSTM	0.719	0.739	0.729

Tabla 4.5: Escenarios de consumo

Modelo	*tnr*	*tpr*	*bacc*
NN_v2	0.930	0.927	0.928
LSTM	0.984	0.949	0.967

Tabla 4.6: Escenarios de generación

Tabla 4.7: Resultados de los detectores *NN_v2* y *LSTM* considerando el *dataset* solar

A partir de este capítulo se pueden extraer varias conclusiones relacionadas con los objetivos del proyecto. Primero, analizando la reproducción de los resultados previos, se puede observar que estos son prácticamente idénticos, menos para la *NN* que mejora la métrica *tnr*. Esto se debe a que el código que se probó fue una versión diferente a la de [2] en la que se le introducía un "sesgo" comentado en [9]. El nuevo objetivo fue mejorar esos resultados eliminando el sesgo para conseguirlos de una forma más avanzada y "real".

Por otro lado, a partir de las tablas anteriores, se puede observar que la gran mayoría de modelos mejoran considerablemente en los escenarios de generación. Esto se puede deber a que los ataques sean más fáciles de detectar, ya que todos los modelos presentan una tendencia similar de un escenario a otro.

Otra conclusión importante es que, sin duda, la red neuronal optimizada mejora el rendimiento del modelo original, de hecho, no hay ningún otro modelo que obtenga unos resultados tan buenos y consistentes como esta. Estos se han conseguido erradicando la principal debilidad del original, que en muchas ocasiones lo hacía incapaz de reconocer los escenarios que no habían sido manipulados.

En último lugar, otro de los objetivos era comprobar si un modelo más sofisticado y complejo, en nuestro caso el modelo *LSTM*, era capaz de ofrecer una mayor precisión que los modelos iniciales. Se puede concluir que indudablemente el modelo mejora los resultados de todos los detectores iniciales menos los de la red neuronal original, a la que solo supera en la detección de los escenarios normales. Por consiguiente, tampoco es capaz de mejorar los resultados del modelo optimizado. Sin embargo, los resultados de este modelo son mucho más consistentes y mucho mejores en algunos casos que los del resto de detectores iniciales, que eran capaces de detectar muy bien ciertos escenarios, mientras que eran incapaces de detectar otros.

Capítulo 5

Metodología y planificación

Este apartado se divide en tres secciones. Inicialmente, en la Sección 5.1 se describe las metodologías de trabajo empleadas en la realización del trabajo. Seguidamente, en la Sección 5.2 se describe la planificación que se ha seguido. Por último, la Sección 5.3 se centra en comentar las horas de dedicación y como estas se han distribuido.

5.1. Metodología

En este apartado se describe tanto la metodología de trabajo empleada durante el desarrollo del proyecto como las distintas herramientas tecnológicas utilizadas.

En primer lugar, como ya se ha comentado en la Sección 4, para la experimentación se ha trabajo con el clúster *"Hermes"* del *I3A* debido a que la ejecución de algunas pruebas podía ser computacionalmente muy costosa, llegando algunas a tardar semanas en finalizar. Este clúster permitió lanzar tantas pruebas como se quisieran simultáneamente sin tener que preocuparse por el consumo de recursos.

Por otro lado, para el desarrollo de nuevos modelos y la realización de algunas pruebas locales, se utilizó Google Colab [16], una plataforma que permite programar y ejecutar código Python [17], el lenguaje de programación utilizado a lo largo de todo el proyecto, en el navegador de manera simple y visual. Además, aquí se crearon algunos programas auxiliares para diferentes tareas. Las distintas herramientas de Python empleadas se pueden ver en el Anexo C.

Finalmente, en cuanto a la metodología, se han seguido algunos principios de las metodologías ágiles [18]. En especial, la realización de reuniones todas las semanas, donde se revisaba el progreso y el estado del trabajo, y se establecían las tareas para la siguiente semana. Además de las reuniones, se mantuvo una comunicación constante a través de correo electrónico para resolver dudas puntuales y mantener a los directores informados sobre el progreso del proyecto.

5.2. Planificación

La planificación del proyecto se dividió en varias etapas, concretamente las enumeradas a continuación:

- **Familiarización con el punto de partida del trabajo**: se hizo especial hincapié en esta fase ya que era fundamental tener una base robusta acerca del contexto del proyecto y del trabajo realizado previamente. Por ello se llevaron a cabo una serie de tareas de estudio del código existente [19], de las diferentes herramientas que se habían utilizado y aquellas con las que se iba a trabajar (ver detalles en Anexo C) y del funcionamiento de las *smart grids*.
- **Reproducción de los resultados previos y validación de los detectores de referencia**: se repitieron los experimentos realizados en [2] con la intención de comprobar que se obtenían los mismos resultados.
- **Preprocesamiento del nuevo conjunto de datos**: en esta fase, se incluye todo lo relacionado con el nuevo conjunto *Solar Ausgrid Dataset* empezando por todas las tareas de preprocesamiento: limpieza, transformación y selección de parámetros relevantes. Y por último, las tareas de creación de los ficheros de entrenamiento y prueba tanto para los escenarios de consumo como los de generación comentados previamente en la Sección 2.1.2.
- **Validación de los detectores *"tegdet"***: esta etapa consiste en evaluar los distintos modelos *teg* con los tres conjuntos de datos disponibles y la comparación de estos con los detectores ya existentes. Todo lo relacionado con esta fase se recoge en las Secciones 3.2 y 4.3. Además, incluye el estudio de [5] y su código fuente [20].
- **Desarrollo del detector *LSTM***: el trabajo de esta etapa se resume en las Secciones 3.3 y 4.4. Incluye el estudio, la implementación, la evaluación y la optimización del modelo.
- **Mejora del detector basado en redes neuronales**: consistió en estudiar la implementación del anterior modelo, probar diferentes tamaños de ventana para calcular la más óptima y optimizar los hiperparámetros. Esta fase se resume en las Secciones 3.4 y 4.4.
- **Pruebas exhaustivas y comparación de los resultados**: se llevaron a cabo diferentes pruebas con los nuevos modelos y se compararon los resultados obtenidos con los modelos anteriores. Estos resultados se resumen en el Capítulo 4 y se exponen de manera más detallada en el Anexo F.

- **Recopilación y análisis de los resultados**: esta etapa es similar a la anterior, en esta se recopilan y analizan los resultados finales (Capítulo 4 y Anexo F) de todas las pruebas y experimentos realizados a lo largo del proyecto.
- **Redacción de la memoria**: esta última etapa consiste en la etapa actual en la que se está elaborando la memoria del proyecto.

En la Figura D.1 del Anexo D se puede encontrar el diagrama de Gantt que refleja visualmente las distintas tareas y su planificación desde el inicio (febrero) hasta el final del proyecto (septiembre).

5.3. Dedicación

La dedicación del proyecto se mide en el número de horas dedicadas al mismo, que, finalmente, han sido un total de 488. Estas se obtiene a partir de la hoja de cálculo usada para el seguimiento de las tareas (Figura D.2). Las distintas categorías consideradas para cada una de las tareas registradas en esta hoja son las siguientes:

- **Reunión**: reuniones entre los miembros (directores y alumno).
- **Estudio**: obtención de conocimientos (artículos, documentación, tecnologías).
- **Pruebas**: desarrollo del código encargado de la verificación y validación.
- **Documentación**: elaboración de la memoria y documentación del código.
- **Implementación**: creación del código.
- **Análisis**: identificación, análisis y resolución de problemas.
- **Gestión**: aquellas tareas que no encajan en ninguna de las categorías anteriores.

El Anexo D proporciona información adicional a esta sección. En él se pueden encontrar las siguientes gráficas: porcentaje del tiempo total invertido en cada categoría (Figura D.3), distribución del tiempo invertido a lo largo de las semanas (Figura D.4), distribución del tiempo invertido a cada categoría a lo largo de las semanas (Figura D.5) y gráfico de cajas del tiempo total invertido en cada categoría (Figura D.6).

Capítulo 6

Conclusiones

En este último capítulo, se presentan las conclusiones extraídas del proyecto. En primer lugar, la Sección 6.1 resume brevemente el resultado del proyecto. En segundo lugar, en la Sección 6.2 se comentan las lecciones aprendidas con la realización de este trabajo. Para finalizar, en la Sección 6.3 se presentan algunos puntos que no han sido abordados en este proyecto, pero podrían ser interesantes para futuras investigaciones.

6.1. Resultado del proyecto

Se considera que los resultados del proyecto han sido satisfactorios porque se han cumplido todos los objetivos que se propusieron al inicio. A continuación, se vuelven a exponer estos objetivos, incluyendo la sección en la que se encuentra la descripción de su cumplimiento.

El primer objetivo fue validar los resultados previos, este se describe detalladamente en la Sección 4.2. Posteriormente, se procedió a evaluar los detectores con un nuevo conjunto de datos, cuyo cumplimiento se aborda en la Sección 4.3. A partir de estos resultados, se definieron los dos últimos objetivos: mejorar la precisión del detector basado en redes neuronales y la implementación de un nuevo modelo que trate de ofrecer una mejor detección que los anteriores. Estos se especifican en la Sección 4.4, además, al final de esta sección se extraen algunas conclusiones respecto a estos objetivos, apoyándose en los resultados de la experimentación.

6.2. Lecciones aprendidas

Durante el desarrollo del proyecto se han aprendido numerosas lecciones. En primer lugar, optar por una metodología ágil para el desarrollo del proyecto fue, sin duda, todo un acierto, ya que facilitó una mayor implicación en el trabajo y minimizó el riesgo de retrasos en las tareas.

En cuanto a los aspectos más técnicos del trabajo, se han adquirido una amplia variedad de conocimientos en técnicas de aprendizaje automático, análisis y minería de datos debido a la necesidad de familiarizarse con las herramientas utilizadas a lo largo del proyecto (ver Anexo C).

Por último, se ha aprendido que, pese a realizar una buena planificación inicial, siempre surgen circunstancias inesperadas durante el desarrollo a las que se les tiene que dedicar más tiempo del deseado, por ello es importante contar con un plan que pueda adaptarse a estas nuevas necesidades a medida que van surgiendo.

6.3. Posibles futuras mejoras

Debido a que el tiempo de desarrollo del proyecto es limitado, hay ciertos aspectos que intencionalmente se han decidido no abordar. Estos se proponen a continuación como posibles futuras mejoras:

- **Utilizar nuevos conjuntos de datos**: sería interesante profundizar en los tipos de redes inteligentes y consumidores mediante el uso de nuevos conjuntos de datos. Esto permitiría evaluar la eficacia de los modelos en una variedad más amplia de situaciones.
- **Desarrollar nuevos escenarios de ataque**: crear nuevos escenarios de ataques relacionados con la generación de energía solar, ya que se ha comprobado que los ataques considerados en el proyecto son más fáciles de detectar que los de los escenarios de consumo. Por tanto, se podrían generar nuevos tipos de ataques para proporcionar una comprensión más completa de la detección de estos.
- **Implementar técnicas de detección de ataques en tiempo real**: incorporar nuevos modelos basados en técnicas de detección en tiempo real o adaptar alguno de los modelos ya existentes, como por ejemplo *LSTM* podría ser un aspecto muy interesante a la vez que complejo. Sin duda sería un gran avance porque podría ser utilizado en un entorno real por compañías eléctricas si se llega a demostrar la capacidad de estos modelos.

Bibliografía

[1] Ahmed S. Musleh, Guo Chen, and Zhao Yang Dong. A survey on the detection algorithms for false data injection attacks in smart grids. *IEEE Transactions on Smart Grid*, 11(3):2218–2234, 2020.

[2] Simona Bernardi, Raúl Javierre, José Merseguer, and José Ignacio Requeno. Detectors of smart grid integrity attacks: an experimental assessment. In *17th European Dependable Computing Conference, EDCC 2021, Munich, Germany, September 13-16, 2021*, pages 75–82. IEEE, 2021.

[3] Commission for energy regulation (CER). `https://www.ucd.ie/issda/data/commissionforenergyregulationcer`, 2023. Accessed: 2023-08-15.

[4] T51318. Ausgrid solar home electricity data notes (august 2014). `https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data`, 2014.

[5] Simona Bernardi, Raúl Javierre, and José Merseguer. tegdet: An extensible python library for anomaly detection using time evolving graphs. *SoftwareX*, 22:101363, 2023.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[7] Varun Badrinath Krishna. *Data-driven methods to improve resource utilization, fraud detection, and cyber-resilience in Smart Grids*. PhD thesis, Graduate College of the University of Illinois at Urbana-Champaign, 2018.

[8] Daisuke Mashima and Alvaro A. Cárdenas. Evaluating Electricity Theft Detectors in Smart Grid Networks. In Davide Balzarotti, Salvatore J. Stolfo, and Marco Cova, editors, *Research in Attacks, Intrusions, and Defenses*, pages 210–229, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[9] Raúl Javierre Cabrero. Detección de ataques de integridad en redes inteligentes mediante el uso de técnicas de aprendizaje automático. Trabajo fin de grado, Universidad de Zaragoza, 2021.

[10] Christopher Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Accessed: 2023-08-15.

[11] A.Mahi al-rashid; F.Hossain; A.Anwar; S.Azam. False data injection attack detection in smart grid using energy consumption forecasting. *Energies*, 15:4877, 2022. https://doi.org/10.3390/en15134877.

[12] AnyScale. What is hyperparameter tuning? https://www.anyscale.com/blog/what-is-hyperparameter-tuning, 2023. Accessed: 2023-08-17.

[13] Geek Culture. How does batch size impact your model learning? https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa, 2023. Accessed: 2023-08-17.

[14] Keras. Keras tuner. https://keras.io/keras_tuner/, 2023. Accessed: 2023-08-17.

[15] Keras. Hyperband tuner. https://keras.io/api/keras_tuner/tuners/hyperband/, 2023. Accessed: 2023-08-17.

[16] Google. Google colaboratory, 2020. Plataforma de desarrollo de código en la nube.

[17] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[18] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.

[19] Diaspore Unizar. smartest. https://github.com/DiasporeUnizar/smartest, 2021. Diaspore GitHub repository.

[20] Diaspore Unizar. Teg. https://github.com/DiasporeUnizar/TEG, 2021. Tegdet GitHub repository.

[21] TensorFlow. Overfit and underfit. https://www.tensorflow.org/tutorials/keras/overfit_and_underfit?hl=es-419, 2023. Accessed: 2023-08-17.

[22] Guillermo Cánovas. tfg-guillermo-canovas. https://github.com/guillecanovas/tfg-guillermo-canovas, 2023. Personal GitHub repository.

[23] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[24] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[25] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*, pages 265–283, 2016.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[28] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[29] François Chollet et al. Keras. https://keras.io, 2015.

[30] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 2020.

[31] François Chollet et al. Keras. `https://keras.io`, 2015.

[32] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[33] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Módulos destacables

Este anexo se utiliza para ver en detalle ciertos módulos que pueden ser interesantes, además, también sirve de guía para llevar a cabo ciertas tareas que, sin una explicación adecuada, podrían resultar complejas. Primero, en el apartado A.1 se explican los pasos necesarios para el lanzamiento de la experimentación final de los diferentes detectores. A continuación, en el apartado A.2 se describen dos opciones disponibles para transformar los datos obtenidos durante la experimentación en formato de tabla. Seguidamente, en A.3 se describen los pasos que se han llevado a cabo para obtener el valor del *threshold* y los distintos módulos encargados de ello. En A.4 se explica el módulo encargado de la búsqueda óptima de hiperparámetros. Después, en el apartado A.5 se expone el código encargado de generar las tablas y gráficas de la comparativa del beneficio económico entre un escenario y otro. Por último, en los apartados A.6 y A.7 se explican los detalles de los detectores *LSTM* y *NN_v2* , respectivamente.

A.1. Lanzamiento de los experimentos

El proceso de lanzamiento de la experimentación implica evaluar cada contador de todos los conjuntos de datos, para cada detector y en cada escenario distinto, tanto los de generación como los de consumo. Todo con el objetivo de obtener no solo la matriz de confusión de cada modelo *(n_tp, n_fp, n_tn, n_fn)*, que es la base para el cálculo de las métricas de calidad asociadas a los detectores, sino también el tiempo que emplean en su creación y en la predicción. Los comandos a ejecutar son los siguientes:

```
1    export PYTHONPATH=.
2    python3 src/experiments/detector_comparer.py <type_of_dataset>
```

```
for meterID in list_of_meterIDs:

    for name_of_detector in list_of_detectors:
        detector = DetectorFactory.create_detector(name_of_detector)
        training_dataset = detector.get_training_dataset(meterID, sys.argv[1], type_of_attack)
        model, time_model_creation = detector.build_model(training_dataset)

        for attack in tuple_of_attacks:
            testing_dataset = detector.get_testing_dataset(attack, meterID, sys.argv[1], type_of_attack)
            predictions, obs, time_model_prediction = detector.predict(testing_dataset, model)
            n_tp, n_tn, n_fp, n_fn = detector.compute_outliers(obs, predictions, attack)

            detector.print_metrics(meterID, name_of_detector, attack, time_model_creation, time_model_prediction, n_tp, n_tn, n_fp, n_fn)
            detector.metrics_to_csv(meterID, name_of_detector, attack, time_model_creation, time_model_prediction, n_tp, n_tn, n_fp, n_fn, type_of_dataset, type_of_attack)

    processed_meterIDs += 1

    remaining_meterIDs = len(list_of_meterIDs) - processed_meterIDs
    avg_time = (time() - t0) / processed_meterIDs

    print(str(remaining_meterIDs) + " meterIDs remaining. It will be completed in " + str(remaining_meterIDs * avg_time) + " seconds (aprox.)")
```

Figura A.1: Código encargado del lanzamiento de los experimentos

A.2. Postprocesamiento de los resultados

En este apartado se describen los pasos necesarios para, a partir del fichero *csv* que se obtienen tras lanzar y esperar a que termine la experimentación, poder ver en formato de tabla los resultados de cada uno de los detectores.

A.2.1. Opción 1

Para esta primera opción, los comandos que se han de ejecutar son los siguientes:

```
export PYTHONPATH=.
python3 src/analytics/__init__.py
python3 src/analytics/dashboradEDCC21.py <type_of_dataset>
```

El resultado son dos ficheros, el primero se muestra en la Figura A.2 y el segundo en la Figura A.3. Las métricas resumidas en el primero se encuentran a continuación, y se puede encontrar más información de las mismas en el Anexo E:

- Accuracy
- Misclass
- Recall
- False Negative Rate (FNR)
- True Negative Rate (TNR)
- False Positive Rate (FPN)
- Precision
- Negative Predictive Value (NPV)
- Balanced accuracy
- F1-score
- Matthews Correlation Coefficient
- Prevalence
- Time build
- Time predict

detector	accuracy	misclass	recall	FNR	TNR	FPN	precision	NPV	balancedAccuracy	f1-score	mcc	prevalence	tb	tp
TEG_Hamming_30	0.444	0.556	0.356	0.644	0.974	0.026	0.988	0.201	0.665	0.524	0.250	0.857	{'min': '0.158', 'mean': '0.327', 'std': '0.168', 'Q1': '0.172', 'Q2': '0.191', 'Q3': '0.509', 'max': '0.868', 'lower-95%': '0.324', 'upper-95%': '0.330', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.087', 'std': '0.046', 'Q1': '0.046', 'Q2': '0.055', 'Q3': '0.130', 'max': '0.300', 'lower-95%': '0.086', 'upper-95%': '0.088', 'CI-half-length': '0.001'}
TEG_Cosine_30	0.308	0.692	0.195	0.805	0.985	0.015	0.988	0.169	0.590	0.325	0.168	0.857	{'min': '0.165', 'mean': '0.338', 'std': '0.176', 'Q1': '0.177', 'Q2': '0.192', 'Q3': '0.529', 'max': '0.997', 'lower-95%': '0.335', 'upper-95%': '0.341', 'CI-half-length': '0.003'}	{'min': '0.040', 'mean': '0.090', 'std': '0.049', 'Q1': '0.047', 'Q2': '0.053', 'Q3': '0.136', 'max': '0.314', 'lower-95%': '0.089', 'upper-95%': '0.090', 'CI-half-length': '0.001'}
TEG_Jaccard_30	0.319	0.681	0.210	0.790	0.968	0.032	0.976	0.170	0.589	0.346	0.161	0.857	{'min': '0.159', 'mean': '0.326', 'std': '0.168', 'Q1': '0.173', 'Q2': '0.186', 'Q3': '0.509', 'max': '0.855', 'lower-95%': '0.324', 'upper-95%': '0.329', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.087', 'std': '0.047', 'Q1': '0.046', 'Q2': '0.051', 'Q3': '0.133', 'max': '0.284', 'lower-95%': '0.087', 'upper-95%': '0.088', 'CI-half-length': '0.001'}
TEG_Dice_30	0.328	0.672	0.221	0.779	0.966	0.034	0.975	0.171	0.594	0.361	0.166	0.857	{'min': '0.160', 'mean': '0.327', 'std': '0.170', 'Q1': '0.173', 'Q2': '0.186', 'Q3': '0.513', 'max': '0.955', 'lower-95%': '0.324', 'upper-95%': '0.330', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.087', 'std': '0.047', 'Q1': '0.045', 'Q2': '0.051', 'Q3': '0.133', 'max': '0.273', 'lower-95%': '0.086', 'upper-95%': '0.088', 'CI-half-length': '0.001'}
TEG_KL_30	0.669	0.331	0.725	0.275	0.330	0.670	0.867	0.167	0.528	0.790	0.043	0.857	{'min': '0.161', 'mean': '0.331', 'std': '0.170', 'Q1': '0.174', 'Q2': '0.189', 'Q3': '0.516', 'max': '0.804', 'lower-95%': '0.328', 'upper-95%': '0.334', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.088', 'std': '0.048', 'Q1': '0.046', 'Q2': '0.052', 'Q3': '0.134', 'max': '0.287', 'lower-95%': '0.087', 'upper-95%': '0.089', 'CI-half-length': '0.001'}
TEG_Jeffreys_30	0.665	0.335	0.625	0.375	0.909	0.091	0.976	0.287	0.767	0.762	0.375	0.857	{'min': '0.161', 'mean': '0.332', 'std': '0.172', 'Q1': '0.174', 'Q2': '0.188', 'Q3': '0.517', 'max': '0.942', 'lower-95%': '0.329', 'upper-95%': '0.335', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.088', 'std': '0.048', 'Q1': '0.046', 'Q2': '0.052', 'Q3': '0.134', 'max': '0.306', 'lower-95%': '0.088', 'upper-95%': '0.089', 'CI-half-length': '0.001'}
TEG_JS_30	0.451	0.549	0.364	0.636	0.970	0.030	0.986	0.203	0.667	0.532	0.251	0.857	{'min': '0.161', 'mean': '0.333', 'std': '0.172', 'Q1': '0.175', 'Q2': '0.189', 'Q3': '0.518', 'max': '0.903', 'lower-95%': '0.330', 'upper-95%': '0.335', 'CI-half-length': '0.003'}	{'min': '0.040', 'mean': '0.089', 'std': '0.048', 'Q1': '0.046', 'Q2': '0.052', 'Q3': '0.135', 'max': '0.286', 'lower-95%': '0.088', 'upper-95%': '0.090', 'CI-half-length': '0.001'}
TEG_Euclidean_30	0.304	0.696	0.191	0.809	0.981	0.019	0.984	0.168	0.586	0.319	0.162	0.857	{'min': '0.159', 'mean': '0.335', 'std': '0.170', 'Q1': '0.173', 'Q2': '0.306', 'Q3': '0.512', 'max': '0.849', 'lower-95%': '0.332', 'upper-95%': '0.338', 'CI-half-length': '0.003'}	{'min': '0.039', 'mean': '0.089', 'std': '0.048', 'Q1': '0.046', 'Q2': '0.068', 'Q3': '0.134', 'max': '0.308', 'lower-95%': '0.088', 'upper-95%': '0.090', 'CI-half-length': '0.001'}

Figura A.2: Fichero ejemplo `solarglobalMetrics.html`

	TEG_Hamming_30	TEG_Cosine_30	TEG_Jaccard_30	TEG_Dice_30	TEG_KL_30	TEG_Jeffreys_30	TEG_JS_30	TEG_Euclidean_30
False	0.974	0.985	0.968	0.966	0.330	0.909	0.970	0.981
RSA_0.25_1.1	0.056	0.010	0.045	0.047	0.814	0.291	0.163	0.020
RSA_0.5_3	0.472	0.047	0.029	0.029	0.955	0.816	0.431	0.028
Avg	0.597	0.802	0.805	0.843	0.811	0.923	0.818	0.841
Swap	0.031	0.019	0.035	0.038	0.696	0.146	0.055	0.025
FDI10	0.639	0.160	0.249	0.265	0.405	0.841	0.329	0.173
FDI30	0.342	0.131	0.098	0.105	0.673	0.731	0.388	0.058

Figura A.3: Fichero ejemplo `solarscenariosAccuracy.html`

A.2.2. Opción 2

Esta segunda opción es algo más sencilla, ya que no requiere tener toda la infraestructura del código y se puede hacer directamente desde el navegador, únicamente se necesita el fichero en formato *csv* que recopila los resultados de la experimentación. Un ejemplo de este fichero se puede ver en la Figura A.4. El código necesario se encuentra en el siguiente enlace: cuaderno de Google Colab

```
meterID,detector,attack,time_model_creation,time_model_prediction,n_tp,n_tn,n_fp,n_fn,accuracy
1540,ARIMAX,False,867.4531064033508,0.156667947769165,0.0,4756.0,284.0,0.0,0.9436507936507936
1540,ARIMAX,RSA_0.5_1.5,867.4531064033508,0.1481571197509765,297.0,0.0,0.0,4743.0,0.0589285714285714
1540,ARIMAX,RSA_0.25_1.1,867.4531064033508,0.1487245559692382,165.0,0.0,0.0,4875.0,0.0327380952380952
1540,ARIMAX,RSA_0.5_3,867.4531064033508,0.1487481594085693,1045.0,0.0,0.0,3995.0,0.2073412698412698
1540,ARIMAX,Avg,867.4531064033508,0.1484789848327636,12.0,0.0,0.0,5028.0,0.0023809523809523
1540,ARIMAX,Min-Avg,867.4531064033508,0.1473057270050048,2310.0,0.0,0.0,2730.0,0.4583333333333333
1540,ARIMAX,Swap,867.4531064033508,0.1475164890289306,823.0,0.0,0.0,4217.0,0.1632936507936508
1540,ARIMAX,FDI0,867.4531064033508,0.1492013931274414,948.0,0.0,0.0,4092.0,0.1880952380952381
1540,ARIMAX,FDI5,867.4531064033508,0.1467673778533935,917.0,0.0,0.0,4123.0,0.1819444444444444
1540,ARIMAX,FDI10,867.4531064033508,0.1487679481506347,774.0,0.0,0.0,4266.0,0.1535714285714285
1540,ARIMAX,FDI20,867.4531064033508,0.1481738090515136,417.0,0.0,0.0,4623.0,0.0827380952380952
1540,ARIMAX,FDI30,867.4531064033508,0.1464931964874267,185.0,0.0,0.0,4855.0,0.0367063492063492
1540,ARIMA,False,55.75068354606629,0.075716495513916,0.0,4776.0,264.0,0.0,0.9476190476190476
1540,ARIMA,RSA_0.5_1.5,55.75068354606629,0.0736758708953857,281.0,0.0,0.0,4759.0,0.0557539682539682
1540,ARIMA,RSA_0.25_1.1,55.75068354606629,0.0744822025299072,99.0,0.0,0.0,4941.0,0.0196428571428571
1540,ARIMA,RSA_0.5_3,55.75068354606629,0.0739474296569824,1041.0,0.0,0.0,3999.0,0.206547619047619
1540,ARIMA,Avg,55.75068354606629,0.0739011764526367,0.0,0.0,0.0,5040.0,0.0
1540,ARIMA,Min-Avg,55.75068354606629,0.0739006996154785,0.0,0.0,0.0,5040.0,0.0
1540,ARIMA,Swap,55.75068354606629,0.0736546516418457,264.0,0.0,0.0,4776.0,0.0523809523809523
1540,ARIMA,FDI0,55.75068354606629,0.0736188888549804,1.0,0.0,0.0,5039.0,0.0001984126984126
1540,ARIMA,FDI5,55.75068354606629,0.0733680725097656,1.0,0.0,0.0,5039.0,0.0001984126984126
1540,ARIMA,FDI10,55.75068354606629,0.0733001232147216,1.0,0.0,0.0,5039.0,0.0001984126984126
1540,ARIMA,FDI20,55.75068354606629,0.0729389190673828,0.0,0.0,0.0,5040.0,0.0
1540,ARIMA,FDI30,55.75068354606629,0.0743472576141357,0.0,0.0,0.0,5040.0,0.0
1540,Min-Avg,False,0.0914876461029052,0.0215787887573242,0.0,15.0,0.0,0.0,1.0
1540,Min-Avg,RSA_0.5_1.5,0.0914876461029052,0.0215167999267578,0.0,0.0,0.0,15.0,0.0
1540,Min-Avg,RSA_0.25_1.1,0.0914876461029052,0.0213096141815185,14.0,0.0,0.0,1.0,0.9333333333333332
1540,Min-Avg,RSA_0.5_3,0.0914876461029052,0.0213043689727783,0.0,0.0,0.0,15.0,0.0
1540,Min-Avg,Avg,0.0914876461029052,0.02134370803833,0.0,0.0,0.0,15.0,0.0
```

Figura A.4: Fichero *csv* que guarda el resultado de la experimentación

```python
import pandas as pd
import io
import numpy as np

dataset = "electricity" #electricity, gas, solar_consumption, solar_generation
detector = ["Min-Avg", "ARIMA", "ARIMAX", "KLD", "JSD", "NN", "NN_v2", "LSTM"]

df = pd.read_csv(dataset + '_detector_comparer_results.csv')

grouped_data = df.groupby(['detector', 'meterID', 'attack'])['accuracy'].mean().reset_index()
pivoted_data = pd.pivot_table(grouped_data, values='accuracy', index='attack', columns='detector', aggfunc='mean')
pivoted_data = pivoted_data.rename_axis(None, axis=0).rename_axis(None, axis=1)

if dataset == 'solar_generation':
  row_order = ["False", "Percentile", "Rating", "RSA_0.5_3"]
else:
  row_order = ["False", "RSA_0.25_1.1", "RSA_0.5_3", "Avg", "Swap", "FDI10", "FDI30"]

html_table = pivoted_data.to_html(border=1, float_format="{:.3f}".format)

file_name = 'table_' + dataset + '_results.html'
with io.open(file_name, 'w', encoding='utf-8') as file:
    file.write(html_table)

print(f"Tabla HTML guardada como '{file_name}'")
```

Figura A.5: Fragmento de código encargado de generar tablas *html* a partir de un fichero *csv*

A.3. Obtención del threshold

El proceso para determinar el valor numérico del *threshold* es un paso crucial en la definición del modelo LSTM, ya que se podría establecer un valor para el umbral muy alto y de esta manera se encontrarían pocas anomalías en los datos, lo que, en teoría, querría decir que el modelo es capaz de clasificar los datos en la categoría a la que corresponden. Sin embargo, si en los datos de *testing* se inyectan valores correspondientes a otros escenarios, estos deberían ser considerados anomalías, pero, al tener el *threshold* tan alto, el modelo es incapaz de detectar estas anomalías reales (falsos negativos). En esta sección se explican los pasos que se han llevado a cabo para poder determinar el valor de este umbral.

A.3.1. Generación de ataques de inyección

La generación de ataques de inyección consiste en tomar 20 contadores aleatorios de cada uno de los conjuntos de datos y en sus ficheros de *testing* de los escenarios normales inyectar ataques en un 10 % aleatorio de las muestras. Esta funcionalidad se recoge en el fichero `generate_attack_injections_file.py` y además también se puede encontrar en el siguiente cuaderno de Google Colab. Para ejecutar correctamente el fichero se han de utilizar los siguientes comandos:

```
export PYTHONPATH=.
python3 src/experiments/generate_attack_injections_file.py <type_of_dataset>
```

A.3.2. Experimentación de valores para el threshold

En este subapartado se explica cómo se utiliza y cómo funciona el *script* encargado de ejecutar experimentos para diferentes umbrales. Sus funcionalidades principales son:

- Definición de diferentes valores para el umbral
- Selección aleatoria de 20 contadores para el experimento
- Ejecución del programa `LSTM_detector_threshold.py` para cada combinación de parámetros y evaluación del modelo guardando los resultados en un fichero *csv* como el de la Figura A.6

Para ejecutar el programa se utilizan los siguientes comandos:

```
export PYTHONPATH=.
python3 src/experiments/threshold_experiments.py <type_of_dataset>
```

Un ejemplo del fichero de salida de este programa se representa en la Figura A.6. Estos ficheros en el código se encuentran en las siguientes rutas del repositorio [22]:

```
script_results/lstm_threshold/lstm_threshold_experiments_electricity.csv
script_results/lstm_threshold/lstm_threshold_experiments_gas.csv
script_results/lstm_threshold/lstm_threshold_experiments_solar_consumption.csv
```

```
meterID,threshold,attack injection,n_tp,n_tn,n_fp,n_fn,accuracy
1871,0.0,FDI10,475,0,4515,0,0.095190380761523
2259,0.0,FDI10,477,0,4513,0,0.0955911823647294
1871,0.0,FDI30,478,0,4512,0,0.0957915831663326
2259,0.0,FDI30,472,0,4518,0,0.0945891783567134
1871,0.05,FDI10,424,663,3852,51,0.2178356713426853
2259,0.05,FDI10,394,1481,3032,83,0.375751503006012
1871,0.05,FDI30,425,636,3876,53,0.212625250501002
2259,0.05,FDI30,401,1394,3124,71,0.3597194388777555
1871,0.1,FDI10,407,1525,2990,68,0.3871743486973948
2259,0.1,FDI10,299,2470,2043,178,0.5549098196392785
1871,0.1,FDI30,351,1103,3409,127,0.2913827655310621
2259,0.1,FDI30,264,2569,1949,208,0.5677354709418838
1871,0.15,FDI10,335,1897,2618,140,0.4472945891783567
2259,0.15,FDI10,239,3108,1405,238,0.6707414829659318
1871,0.15,FDI30,319,1846,2666,159,0.4338677354709419
2259,0.15,FDI30,233,3029,1489,239,0.6537074148296593
1871,0.25,FDI10,275,2727,1788,200,0.6016032064128256
2259,0.25,FDI10,153,3633,880,324,0.7587174348697395
1871,0.25,FDI30,268,2790,1722,210,0.6128256513026052
2259,0.25,FDI30,176,3623,895,296,0.7613226452905811
1871,0.3,FDI10,239,3033,1482,236,0.6557114228456914
2259,0.3,FDI10,129,3794,719,348,0.7861723446893788
1871,0.3,FDI30,247,3011,1501,231,0.6529058116232465
2259,0.3,FDI30,130,3797,721,342,0.7869739478957916
1871,0.35,FDI10,222,3254,1261,253,0.6965931863727455
2259,0.35,FDI10,113,3943,570,364,0.8128256513026052
1871,0.35,FDI30,215,3307,1205,263,0.7058116232464929
2259,0.35,FDI30,115,3913,605,357,0.8072144288577154
1871,0.4,FDI10,202,3483,1032,273,0.7384769539078156
2259,0.4,FDI10,100,4003,510,377,0.8222444889779559
```

Figura A.6: Fichero *csv* de salida

A.3.3. Generación de gráficas para determinar el valor

Este programa se encarga de analizar los resultados de los experimentos realizados para detectar anomalías. El principal objetivo es observar cómo diferentes valores del *threshold* afectan la tasa de falsos negativos y la precisión del modelo. El programa está representado en la Figura A.7 y el resultado del mismo en la Figura 3.5. También se puede ver el código en el siguiente cuaderno de Google Colab.

```python
import pandas as pd
import os
import numpy as np
from matplotlib import pyplot as plt

df = pd.read_csv('lstm_threshold_experiments.csv')

filtered_df = df[(df['threshold'] < 1.75) | (df['threshold'] > 5)]

filtered_df_none = filtered_df[(filtered_df['attack injection'] == "None")]
filtered_df_attack = filtered_df[(filtered_df['attack injection'] != "None")]

grouped_data_none = filtered_df_none.groupby(['threshold']).mean()
grouped_data_attack = filtered_df_attack.groupby(['threshold']).mean()

mean_n_fn = grouped_data_attack['n_fn']
mean_accuracy_attack = grouped_data_attack['accuracy']
mean_accuracy_none = grouped_data_none['accuracy']

mean_n_fn_percentage = (mean_n_fn * 100) / 700
mean_accuracy_attack_percentage = mean_accuracy_attack * 100
mean_accuracy_none_percentage = mean_accuracy_none * 100

threshold_values = mean_n_fn.index

plt.figure(figsize=(7, 4))
plt.plot(threshold_values, mean_n_fn_percentage, label='False negatives rate')
plt.plot(threshold_values, mean_accuracy_attack_percentage, label='Accuracy (Attack Injection)')
plt.plot(threshold_values, mean_accuracy_none_percentage, label='Accuracy (No Injections)')

plt.xlabel('Threshold value')
plt.ylabel('Rate (%)')
plt.legend()
plt.grid(True)

output_file = "threshold.png"
plt.savefig(output_file)
plt.show()
```

Figura A.7: Fragmento de código encargado de generar la gráfica para determinar el threshold

A.4. Lanzamiento *tuner* de hiperparámetros

Se tienen dos programas encargados de ejecutar la optimización de hiperparámetros, uno para el modelo *NN_v2* (red neuronal optimizada) y otro para el modelo *LSTM*. Las principales funcionalidades de estos programas son, por un lado, seleccionar aleatoriamente 10 contadores correspondientes del conjunto de datos seleccionado y, por otro lado, ejecutar el programa `hyperparameter_tuning_<NN,LSTM>.py` para cada contador con los parámetros correspondientes. Los comandos necesarios para ejecutar los programas son:

```
1    export PYTHONPATH=.
2    python3 src/experiments/run_hyperparameter_tuning_NN.py <type_of_dataset>
```

```
1    export PYTHONPATH=.
2    python3 src/experiments/run_hyperparameter_tuning_LSTM.py <type_of_dataset>
```

Como se ha comentado, los programas encargados de la ejecución de la optimización son: `hyperparameter_tuning_NN.py` e `hyperparameter_tuning_LSTM.py`. Estos proporcionan la funcionalidad de la red neuronal optimizada y del modelo *LSTM*, respectivamente y se centran en buscar la mejor combinación de hiperparámetros para ambos modelos, a tal efecto se utiliza la biblioteca *Keras Tuner* [14].

```python
import subprocess
import sys
from src import meterIDsGas, meterIDsElectricity, meterIDsSolar
import random


if __name__ == '__main__':

    """
    args:
    sys.argv[1]: type of dataset (electricity, gas, solar_consumption, solar_generation)
    """

    if len(sys.argv) != 2:
        print("Usage: python3 run_hyperparameter_tuning_NN.py <electricity/gas/solar_consumption/solar_generation")
        exit(85)

    type_of_dataset = sys.argv[1]

    if type_of_dataset == "electricity":
        list_of_meterIDs = meterIDsElectricity
    elif type_of_dataset == "gas":
        list_of_meterIDs = meterIDsGas
    elif type_of_dataset == "solar_consumption" or type_of_dataset == "solar_generation":
        list_of_meterIDs = meterIDsSolar
    else:
        print("Usage: python3 run_hyperparameter_tuning.py <electricity/gas/solar_consumption/solar_generation")
        exit(85)


    # Randomly select 10 meter IDs
    list_of_meterIDs = random.sample(list_of_meterIDs, 10)


    # Call the command with each combination as parameters
    for meter_id in list_of_meterIDs:
        command = f"python3 src/experiments/hyperparameter_tuning_NN.py {type_of_dataset} {meter_id}"
        subprocess.run(command, shell=True)


```

Figura A.8: Programa `hyperparameter_tuning_NN.py`

```python

hp = HyperParameters()
hp.Int("n_layers", min_value = 0 , max_value = 4)
hp.Int("batch_size", min_value = 8, max_value = 256, step = 16)
hp.Int("neuron_units", min_value = 32, max_value = 512, step = 32)

hyperband_tuner = Hyperband(
                    MyHyperModel(),
                    hyperparameters = hp,
                    objective = "val_accuracy",
                    project_name ="hyperband_NN_tuning_" + str(WINDOW_SIZE) + '_' + str(meterID),
                    max_epochs = 10,
                    hyperband_iterations = 1,
                    directory = "main_dir",
                    overwrite = True,
                )

early_stopping = EarlyStopping(monitor='val_accuracy', verbose = 0, patience = 10, min_delta = 1e-3, restore_best_weights = True)
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy', save_best_only=True, save_weights_only=False, mode='min', verbose=0)

hyperband_tuner.search( x,
                        y,
                        steps_per_epoch = None,
                        validation_split = 0.2,
                        verbose = 1, #
                        #callbacks = [checkpoint],
                        use_multiprocessing = True,
                    )

hyperband_best_hps = hyperband_tuner.get_best_hyperparameters()[0]
hyperband_best_model = hyperband_tuner.hypermodel.build(hyperband_best_hps)

with open('./script_results/tuner_results/NN/tuner_nn_results' + '.txt', 'a') as f:
    print("NN (meter_id = " + str(meterID) + ", window = " + str(WINDOW_SIZE) + ", dataset = " + str(type_of_dataset) + ")", file=f)
    print(hyperband_best_hps.values, file=f)
    print("================================================", file=f)


hyperband_best_hps.values
```

Figura A.9: Fragmento de código encargado de la búsqueda de los hiperparámetros

A.5. Calcular el beneficio de los ataques

En este apartado se da acceso al código utilizado para la generación de las Tablas 2.1 y 2.2 y de las Figuras 2.1 y 2.2. El código se encuentra en el siguiente cuaderno de Google Colab.

A.6. Modelo LSTM

En este apartado se da acceso al cuaderno de Google Colab en el que se puede probar localmente el modelo *LSTM* para un contador en un escenario concreto. El ejemplo que aparece en el cuaderno es utilizando el contador cuyo identificador es 1014 del conjunto de datos de electricidad para el escenario normal. Se puede acceder al cuaderno mediante el siguiente enlace: modelo *LSTM* en Google Colab.

A.7. Red Neuronal optimizada

Este apartado tiene la misma función que el anterior pero, en este caso, con el modelo de la red neuronal optimizada. Se puede acceder al cuaderno mediante el siguiente enlace: modelo *NN_v2* en Google Colab

Anexos B

Conjuntos de datos

En este anexo se detalla los conjuntos de datos utilizados: electricidad (B.1), gas (B.2) y solar (B.3). Se describen las características de estos, incluyendo sus dimensiones, fuentes y estructuras. Además, a continuación se detallan los comandos a ejecutar para generar los ficheros de *training* y *testing* para cada uno de los *datasets*.

```
export PYTHONPATH=.
python3 src/preprocessing/training_and_testing_generator.py <type_of_dataset>
```

```
for meterID in meterIDs:
    attack_injector = AttackInjector(caseID=meterID)

    # Training weeks
    training_set = dataset_training_all_meter_ids.query('ID == @meterID')

    if dataset.lower() == "solar_generation":

        print("\nGenerating Training " + dataset.lower() + " file:", SCRIPT_RESULTS + dataset.lower() + "_training_data/" + str(meterID) + "_" + str(first_week_train) + "_" + str(last_week_train) + ".csv")
        training_set.to_csv(SCRIPT_RESULTS + dataset.lower() + '_' + type.lower() + "_training_data/" + str(meterID) + "_" + str(first_week_train) + "_" + str(last_week_train) + ".csv", index=False)

        generate_attacked_file(attack_injector, meterID, training_set,  dataset.lower(), first_week_train, last_week_train, "training_data", "Rating")
        generate_attacked_file(attack_injector, meterID, training_set,  dataset.lower(), first_week_train, last_week_train, "training_data", "Percentile")
        generate_attacked_file(attack_injector, meterID, training_set,  dataset.lower(), first_week_train, last_week_train, "training_data", "RSA_0.5_3", 0.5, 3)


    else:
        print("\nGenerating Training " + dataset.lower() +  " file:", SCRIPT_RESULTS + dataset.lower() + "_training_data/" + str(meterID) + "_" + str(first_week_train) + "_" + str(last_week_train) + ".csv")
        training_set.to_csv(SCRIPT_RESULTS + dataset.lower() + "_training_data/" + str(meterID) + "_" + str(first_week_train) + "_" + str(last_week_train) + ".csv", index=False)

        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "Swap")
        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "Avg", 0.5, 1.5)
        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "FDI10")
        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "FDI30")
        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "RSA_0.25_1.1", 0.25, 1.1)
        generate_attacked_file(attack_injector, meterID, training_set, dataset.lower(), first_week_train, last_week_train, "training_data", "RSA_0.5_3", 0.5, 3)
```

Figura B.1: Fragmento de código para generar ficheros de entrenamiento

```
# Testing weeks
testing_set = dataset_testing_all_meter_ids.query('ID == @meterID')

if dataset.lower() == "solar_generation":

    print("Generating Testing " + dataset.lower() +  "file:", SCRIPT_RESULTS + dataset.lower() + "_testing_data/" + str(meterID) + "_" + str(first_week_test) + "_" + str(last_week_test) + ".csv")
    testing_set.to_csv(SCRIPT_RESULTS + dataset.lower() + '_' + type.lower() + "_testing_data/" + str(meterID) + "_" + str(first_week_test) + "_" + str(last_week_test) + ".csv", index=False)

    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "Rating")
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "Percentile")
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "RSA_0.5_3", 0.5, 3)


else:
    print("Generating Testing " + dataset.lower() +  " file:", SCRIPT_RESULTS + dataset.lower() + "_testing_data/" + str(meterID) + "_" + str(first_week_test) + "_" + str(last_week_test) + ".csv")
    testing_set.to_csv(SCRIPT_RESULTS + dataset.lower() + "_testing_data/" + str(meterID) + "_" + str(first_week_test) + "_" + str(last_week_test) + ".csv", index=False)

    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "Swap")
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "Avg", 0.5, 1.5)
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "FDI10")
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "FDI30")
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "RSA_0.25_1.1", 0.25, 1.1)
    generate_attacked_file(attack_injector, meterID, testing_set, dataset.lower(), first_week_test, last_week_test, "testing_data", "RSA_0.5_3", 0.5, 3)
```

Figura B.2: Fragmento de código para generar ficheros de prueba

Tras este proceso, se crean los ficheros de *training* y *testing* cuya sintaxis es "`<id>_<escenario>_inicio_fin.csv`" (p.ej.: *1014_Avg_0_50.csv*) siendo id el identificador del contador, escenario el nombre del ataque o vacío si es un escenario normal y los valores 'inicio' y 'fin' hacen referencia a que el fichero guarda los datos desde la semana *inicio* hasta la semana *fin*, los valores que toman estas variables dependen del conjunto de datos, estos se detallan en la Tabla B.1. El contenido final de los ficheros se ejemplifica en las Figuras B.3 (Electricidad), B.4 (Gas), B.6 (Solar consumo) y B.7 y B.8 (Solar generación):

Conjunto de datos	**Semanas de *training***	**Semanas de *testing***
Electricidad *(ISSDA CER)*	0 - 50	51 - 75
Gas *(ISSDA CER)*	0 - 60	61 - 77
Solar *(Ausgrid)*	0 - 50	51 - 101

Tabla B.1: Semanas de inicio y finalización de los datos de entrenamiento y prueba para los distintos *datasets*

B.1. Electricidad (ISSDA CER)

Tanto este conjunto de datos como el correspondiente a la subsección B.2, se obtienen del *Comisión for Energy Regulation (CER)*, más concretamente del *Irish Social Science Data Archive (ISSDA)*. Para obtener acceso, es necesario enviar una solicitud de uso al *ISSDA* que es aceptada únicamente para propósitos de investigación o educativos.

El *dataset* en cuestión contiene datos del uso de la energía eléctrica de 6445 contadores en Irlanda durante varias semanas, con muestreos cada 30 minutos. Sin embargo, por motivos del coste computacional de la experimentación, se utilizaron únicamente 1000, asegurándose que estos tuviesen una lectura cada media hora para todas las semanas. Estos contadores pueden pertenecer las siguientes tres categorías:

Tipo de contador	**Descripción**
Residential	Hogar
SME	Pequeñas y medianas empresas
Unclassified	No clasificados

Tabla B.2: Descripción del tipo de contador

En cuanto al preprocesado de los datos, este se lleva a cabo en el fichero cuya ruta es: `src/preprocessing/isddacer.py` [22]. Sin embargo, este aspecto no fue objeto de nuestro proyecto, ya que había sido abordado en [9]. En este trabajo se partía directamente de los ficheros de entrenamiento y prueba, tal y como se muestran en la siguiente figura:

```
ID,DT,Usage
1014,19501,0.204
1014,19502,0.39299999999999996
1014,19503,0.21
1014,19504,0.153
1014,19505,0.18100000000000002
1014,19506,0.196
1014,19507,0.152
1014,19508,0.46299999999999997
1014,19509,0.295
1014,19510,0.20800000000000002
1014,19511,0.271
```

(a) Normal

```
ID,DT,Usage
1014,19501,0.9624413592197837
1014,19502,0.9850099056150944
1014,19503,0.8003820197462221
1014,19504,0.5270082989808212
1014,19505,0.45321854348309715
1014,19506,0.9854699964880774
1014,19507,0.7905478325748981
1014,19508,0.950047427802856
1014,19509,0.4982156458311336
1014,19510,0.6168258708269372
1014,19511,0.8136245906497132
```

(b) Avg

```
ID,DT,Usage
1014,19501,0.0204
1014,19502,0.0393
1014,19503,0.021
1014,19504,0.015300000000000001
1014,19505,0.0181
1014,19506,0.019600000000000003
1014,19507,0.0152
1014,19508,0.0463
1014,19509,0.0295
1014,19510,0.020800000000000003
1014,19511,0.027100000000000003
```

(c) FDI10

```
ID,DT,Usage
1014,19501,0.06119999999999999
1014,19502,0.11789999999999998
1014,19503,0.063
1014,19504,0.045899999999999996
1014,19505,0.05430000000000001
1014,19506,0.0588
1014,19507,0.045599999999999995
1014,19508,0.1389
1014,19509,0.0885
1014,19510,0.062400000000000004
1014,19511,0.0813
```

(d) FDI30

```
ID,DT,Usage
1014,19501,0.08373713845243755
1014,19502,0.2940249157395330 4
1014,19503,0.14176500416635282
1014,19504,0.07183644325488708
1014,19505,0.1628775588268414
1014,19506,0.05888805777629783
1014,19507,0.05485756194945655
1014,19508,0.31581297082677673
1014,19509,0.30218765739388864
1014,19510,0.22351685995551676
1014,19511,0.07185575286911433
```

(e) RSA 0.25 1.1

```
ID,DT,Usage
1014,19501,0.2406356845515527
1014,19502,0.8641580787789231
1014,19503,0.5168817068539746
1014,19504,0.2519248338150938
1014,19505,0.10144535382493382
1014,19506,0.5403517688433807
1014,19507,0.10188446631023282
1014,19508,0.6551268200812381
1014,19509,0.2999137096913125
1014,19510,0.5356434439876006
1014,19511,0.4877492533991643
```

(f) RSA 0.5 3

```
ID,DT,Usage
1014,19501,3.603
1014,19502,2.937
1014,19503,2.923
1014,19504,2.917
1014,19505,2.4219999999999997
1014,19506,2.16
1014,19507,1.376
1014,19508,1.237
1014,19509,1.137
1014,19510,1.078
1014,19511,0.951
```

(g) Swap

Figura B.3: Ficheros de *training* de los distintos escenarios para el contador #1014

B.2. Gas (ISSDA CER)

En este *dataset* el único cambio relevante respecto al de la electricidad es el tamaño, ya que contiene información únicamente de 1576 contadores, de todos modos, finalmente del total se seleccionan 1000 en los que se asegura que estos tengan un registro cada media hora para todas las semanas.

En cuanto al preprocesado de los datos, este también se lleva a cabo en el fichero cuya ruta es la siguiente: `src/preprocessing/isddacer.py` [22] aunque este es algo más sencillo debido a que inicialmente los datos ya están distribuidos en semanas. Sin embargo, igual que en el apartado anterior, este aspecto no fue objeto de nuestro proyecto. Se puede ver a continuación un ejemplo de los ficheros de entrenamiento y prueba con los que se trabaja en este conjunto de datos.

```
ID,DT,Usage
1000,33501,0.893845063291139
1000,33502,0.607975696202532
1000,33503,0.684475949367089
1000,33504,0.8173448101265821
1000,33505,0.607975696202532
1000,33506,0.849555443037975
1000,33507,0.6633377215189871
1000,33508,0.6069691139240501
1000,33509,0.9059240506329108
1000,33510,0.6069691139240501
1000,33511,0.607975696202532
```

(a) Normal

```
ID,DT,Usage
1000,33501,2.189622023532987
1000,33502,3.049419190399248
1000,33503,2.6282409633405606
1000,33504,3.343376015272634
1000,33505,2.919927165600067
1000,33506,2.4175170700393216
1000,33507,3.640487607161981
1000,33508,3.9165781064113907
1000,33509,1.8059146982421108
1000,33510,2.4326475794819244
1000,33511,4.076771221899989
```

(b) Avg

```
ID,DT,Usage
1000,33501,0.0893845063291139
1000,33502,0.060797569620253206
1000,33503,0.0684475949367089
1000,33504,0.08173448101265822
1000,33505,0.060797569620253206
1000,33506,0.08495554430379751
1000,33507,0.06633377215189872
1000,33508,0.06069691139240501
1000,33509,0.09059240506329108
1000,33510,0.06069691139240501
1000,33511,0.060797569620253206
```

(c) FDI10

```
ID,DT,Usage
1000,33501,0.2681535189873417
1000,33502,0.1823927088607596
1000,33503,0.2053427848101267
1000,33504,0.24520344303797462
1000,33505,0.1823927088607596
1000,33506,0.2548666329113925
1000,33507,0.1990013164556961
1000,33508,0.18209073417721502
1000,33509,0.27177721518987324
1000,33510,0.18209073417721502
1000,33511,0.1823927088607596
```

(d) FDI30

```
ID,DT,Usage
1000,33501,0.7597075513247964
1000,33502,0.4260974720312493
1000,33503,0.555985311914196
1000,33504,0.7504965368205145
1000,33505,0.5671173535318169
1000,33506,0.46426357989409806
1000,33507,0.48291023204345923
1000,33508,0.40523097524287355
1000,33509,0.8774442587861293
1000,33510,0.2979568420081548
1000,33511,0.2082402255599413
```

(e) RSA 0.25 1.1

```
ID,DT,Usage
1000,33501,1.606290983029685
1000,33502,0.3964154110340588
1000,33503,2.0416410479831195
1000,33504,1.932369148843718
1000,33505,1.775236289182536
1000,33506,1.7026558095807345
1000,33507,0.40534230021421985
1000,33508,1.5552386411887267
1000,33509,1.489450002092367
1000,33510,0.37088735552662955
1000,33511,1.6170539399324362
```

(f) RSA 0.5 3

```
ID,DT,Usage
1000,33501,13.3855311392405
1000,33502,12.5027584810127
1000,33503,11.265668860759499
1000,33504,11.2314450632911
1000,33505,10.459396455696199
1000,33506,10.3043827848101
1000,33507,10.127224303797501
1000,33508,9.8182035443038
1000,33509,8.316382784810129
1000,33510,5.48889316455696
1000,33511,4.9926481012658215
```

(g) Swap

Figura B.4: Ficheros de *training* de los distintos escenarios para el contador #1000

B.3. Energía solar (Ausgrid)

Este conjunto de datos proviene de una de las principales empresas de distribución de electricidad en Australia, *Ausgrid*, que ha decidido hacerlo público "con la intención de que sea utilizado por organizaciones e individuos con diversos fines, incluida la investigación, la formulación de políticas y el suministro de información sobre el rendimiento de los sistemas solares" [4].

El *dataset* contiene datos de 300 clientes con instalaciones de paneles solares. Estos datos recogidos van desde el 1 de julio de 2010 hasta el 30 de junio de 2012. Los ficheros obtenidos son ficheros en formato *csv*, uno para cada año, con información de las siguientes columnas:

Campos	**Descripción**
Customer	ID del cliente que va desde 1 a 300
Postcode	Código postal del cliente
Generator Capacity	Capacidad o potencia máxima del panel solar (kWp)
Consumption Category	**GC**: consumo general excluyendo la generación solar y el suministro de carga controlada; **CL**: Consumo de carga controlada; **GG**: Generación bruta generada por el panel solar
Date	Fecha en formato *DDMMMYYYY*
0:30-23:30	48 columnas que representan la energía consumida o generada desde la última lectura, una por cada media hora
Row Quality	Indica la calidad de los datos, en blanco si cada valor de cada media hora es el registro real o **NA** si alguno de los valores se basan en estimaciones

Tabla B.3: Descripción del formato de los datos

El trabajo llevado a cabo para transformar estos ficheros en formato *csv* en lo que se puede ver en las Figuras B.6, B.7 y B.8, se recoge en el fichero `src/preprocessing/ausgrid.py` [22]. Hay que tener en cuenta que en este conjunto de datos, el trabajo de preprocesado se divide en dos categorías, una para cada escenario: consumo y generación. Este proceso se describe con mayor detalle en las subsecciones B.3.1 y B.3.2 respectivamente.

B.3.1. Escenarios de consumo

Se trabajó con dos ficheros *2010-2011 Solar home electricity data.csv* y *2011-2012 Solar home electricity data v2.csv*, con el primero de los dos no hubo ningún problema, se recorrió el fichero comprobando si cada vivienda tenía solo una tarifa principal (*Consumption Category* = 'GC') o también contaba con una tarifa de carga controlada (*Consumption Category* = 'CL'), es decir, si tenía una fila con 'GC' y otra con 'CL' o solo tenía la primera.

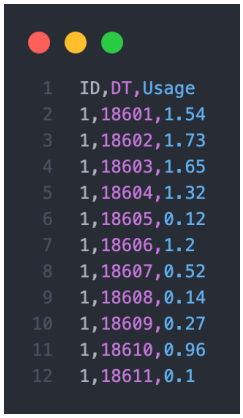
En caso de contar con las dos tarifas, entonces los registros se obtienen calculando la suma de cada media hora para la fila con 'GC' y para la fila con 'CL', en caso de que solo tenga la primera entonces la energía consumida en cada media hora se obtiene directamente de la misma fila sin necesidad de sumar.

Es decir, para la obtención de un fichero con el mismo formato que los de la Figura B.6 se han de seguir los siguientes pasos. En primer lugar, la columna *Usage* se consigue colocando cada media hora del día del fichero inicial en una línea nueva del fichero que se quiere generar. Seguidamente, la columna *ID* tendrá el valor del identificador de la vivienda, es decir, el valor de la columna *Customer*. Por último, para obtener la columna DT se hace uso de una función muy simple que calcula el día del año correspondiente al valor de la columna *Date*, este valor corresponderá a los tres primeros dígitos de la columna, mientras que los dos últimos se calculan a partir de la media hora del día, al haber 48 medias horas, estos dos dígitos toman valores desde 01 hasta 48.

El segundo de los ficheros para el preprocesamiento sigue el mismo funcionamiento y tiene exactamente la misma estructura pero tenía un pequeño problema de inconsistencias en los datos. Había contadores que pese a tener las dos tarifas contratadas, en algunas fechas determinadas solo se registraba la tarifa principal. Esas fechas tuvieron que ser identificadas para que a la hora de hacer la suma de lo consumido en ambas tarifas se sumase directamente un 0 para el valor de la tarifa 'CL'. El código encargado de esto se muestra en la Figura B.5

```python
def missing_rows(file):
  df_orig = pd.read_csv(file, skiprows=1) # Lee el DataFrame original
  cols = ['Customer','date']
  df_orig = df_orig[df_orig['Consumption Category'] == 'CL']
  df_orig = df_orig[cols]

  ids = df_orig['Customer'].drop_duplicates() # Genera todas las combinaciones posibles de ID y Date

  dates = pd.date_range(start='07/01/2011', end='06/30/2012')
  dates_str = dates.strftime('%-d/%m/%Y').tolist()

  combinations = list(itertools.product(ids, dates_str))

  df_all = pd.DataFrame(combinations, columns=['Customer', 'date']) # Crea un nuevo DataFrame con las combinaciones

  # Encuentra las combinaciones que faltan
  df_missing = pd.merge(df_all, df_orig, on=['Customer', 'date'], how='outer', indicator=True)
  df_missing = df_missing[df_missing['_merge'] == 'left_only'].drop('_merge', 1)

  half_hour_columns = pd.date_range(start='0:00', end='23:30', freq='30min').strftime('%-H:%M').tolist()
  half_hour_columns.append(half_hour_columns.pop(0))

  new_df = pd.DataFrame(0, index=df_missing.index, columns=half_hour_columns)
  df_missing = pd.concat([df_missing, new_df], axis=1)

  return df_missing
```

Figura B.5: Código encargado de encontrar las filas restantes de ‘CL’



```
ID,DT,Usage
1,18601,1.54
1,18602,1.73
1,18603,1.65
1,18604,1.32
1,18605,0.12
1,18606,1.2
1,18607,0.52
1,18608,0.14
1,18609,0.27
1,18610,0.96
1,18611,0.1
```

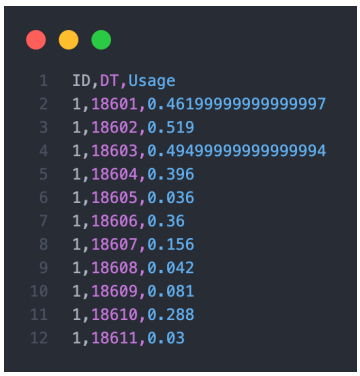
(a) Normal

```
ID,DT,Usage
1,18601,0.4696065394362549
1,18602,0.656237619626535
1,18603,0.9911316710001631
1,18604,0.48754627985078647
1,18605,0.8169081878345728
1,18606,1.017259433349602
1,18607,0.7481963597678583
1,18608,0.3926902807305884
1,18609,0.9815780792479293
1,18610,0.5916959888518049
1,18611,0.8399598121676229
```

(b) Avg

```
ID,DT,Usage
1,18601,0.15400000000000003
1,18602,0.17300000000000001
1,18603,0.165
1,18604,0.132
1,18605,0.012
1,18606,0.12
1,18607,0.052000000000000005
1,18608,0.014000000000000002
1,18609,0.027000000000000003
1,18610,0.096
1,18611,0.010000000000000002
```

(c) FDI10



```
ID,DT,Usage
1,18601,0.46199999999999997
1,18602,0.519
1,18603,0.49499999999999994
1,18604,0.396
1,18605,0.036
1,18606,0.36
1,18607,0.156
1,18608,0.042
1,18609,0.081
1,18610,0.288
1,18611,0.03
```

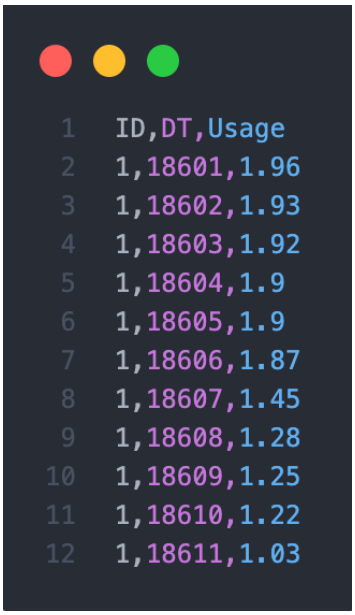
(d) FDI30

```
ID,DT,Usage
1,18601,0.8890950324116303
1,18602,0.6725292469236298
1,18603,0.9989210507167476
1,18604,0.9950182134491595
1,18605,0.09927477662706004
1,18606,0.8307329466015734
1,18607,0.3641246547467406
1,18608,0.04214773529019157
1,18609,0.21926449888479793
1,18610,0.3039405968713446
1,18611,0.06591884353001376
```

(e) RSA 0.25 1.1

```
ID,DT,Usage
1,18601,4.586383324977387
1,18602,3.7666228335119274
1,18603,2.897151250474505
1,18604,2.328532204387426
1,18605,0.3525215671600123
1,18606,0.8922109388066511
1,18607,1.3845784612094456
1,18608,0.24744634532293855
1,18609,0.7244600754235597
1,18610,1.289358416066762
1,18611,0.21727896412696346
```

(f) RSA 0.5 3



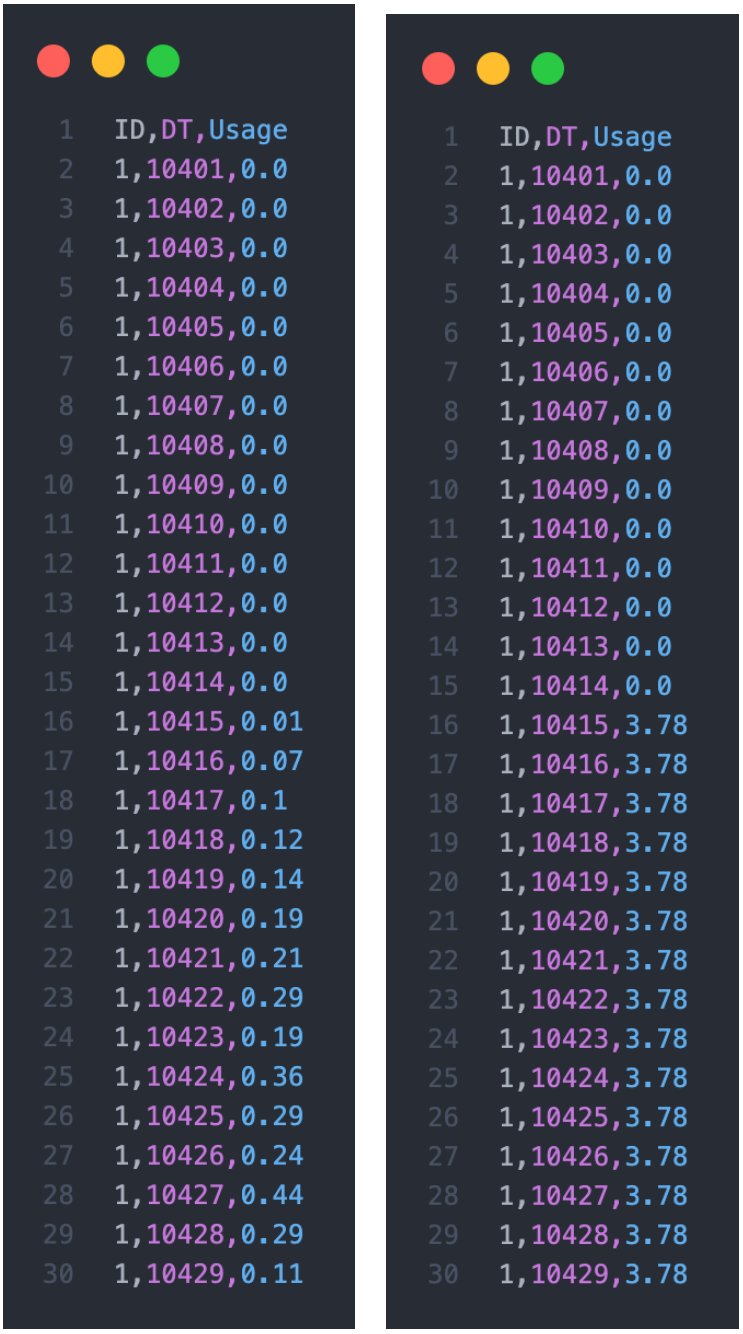
```
ID,DT,Usage
1,18601,1.96
1,18602,1.93
1,18603,1.92
1,18604,1.9
1,18605,1.9
1,18606,1.87
1,18607,1.45
1,18608,1.28
1,18609,1.25
1,18610,1.22
1,18611,1.03
```

(g) Swap

Figura B.6: Ficheros de *training* de los distintos escenarios de consumo para el contador #1

B.3.2. Escenarios de generación

Para el escenario de generación se sigue el mismo procedimiento que en el escenario de consumo para las viviendas que solo tenían una de las dos tarifas. La única diferencia es que se ha de coger las filas cuya columna *Consumption Category* sea igual a 'GG' e ir registrando el valor para cada media hora del fichero, de esta manera, como se ha explicado con anterioridad, se consiguen directamente los valores para la columna *Usage* de los ficheros finales de entrenamiento y prueba. Las columnas DT e ID se obtienen exactamente igual que antes.



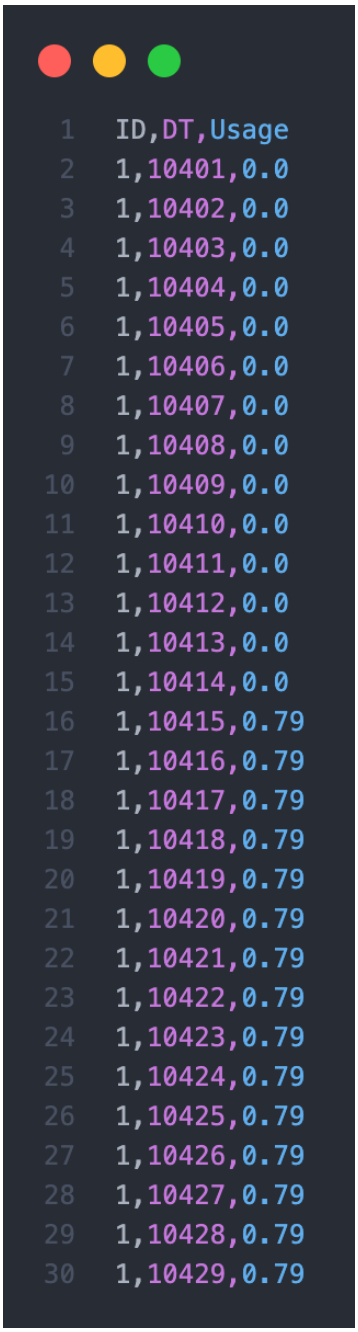
```
ID,DT,Usage
1,10401,0.0
1,10402,0.0
1,10403,0.0
1,10404,0.0
1,10405,0.0
1,10406,0.0
1,10407,0.0
1,10408,0.0
1,10409,0.0
1,10410,0.0
1,10411,0.0
1,10412,0.0
1,10413,0.0
1,10414,0.0
1,10415,0.01
1,10416,0.07
1,10417,0.1
1,10418,0.12
1,10419,0.14
1,10420,0.19
1,10421,0.21
1,10422,0.29
1,10423,0.19
1,10424,0.36
1,10425,0.29
1,10426,0.24
1,10427,0.44
1,10428,0.29
1,10429,0.11
```

(a) Normal

```
ID,DT,Usage
1,10401,0.0
1,10402,0.0
1,10403,0.0
1,10404,0.0
1,10405,0.0
1,10406,0.0
1,10407,0.0
1,10408,0.0
1,10409,0.0
1,10410,0.0
1,10411,0.0
1,10412,0.0
1,10413,0.0
1,10414,0.0
1,10415,3.78
1,10416,3.78
1,10417,3.78
1,10418,3.78
1,10419,3.78
1,10420,3.78
1,10421,3.78
1,10422,3.78
1,10423,3.78
1,10424,3.78
1,10425,3.78
1,10426,3.78
1,10427,3.78
1,10428,3.78
1,10429,3.78
```

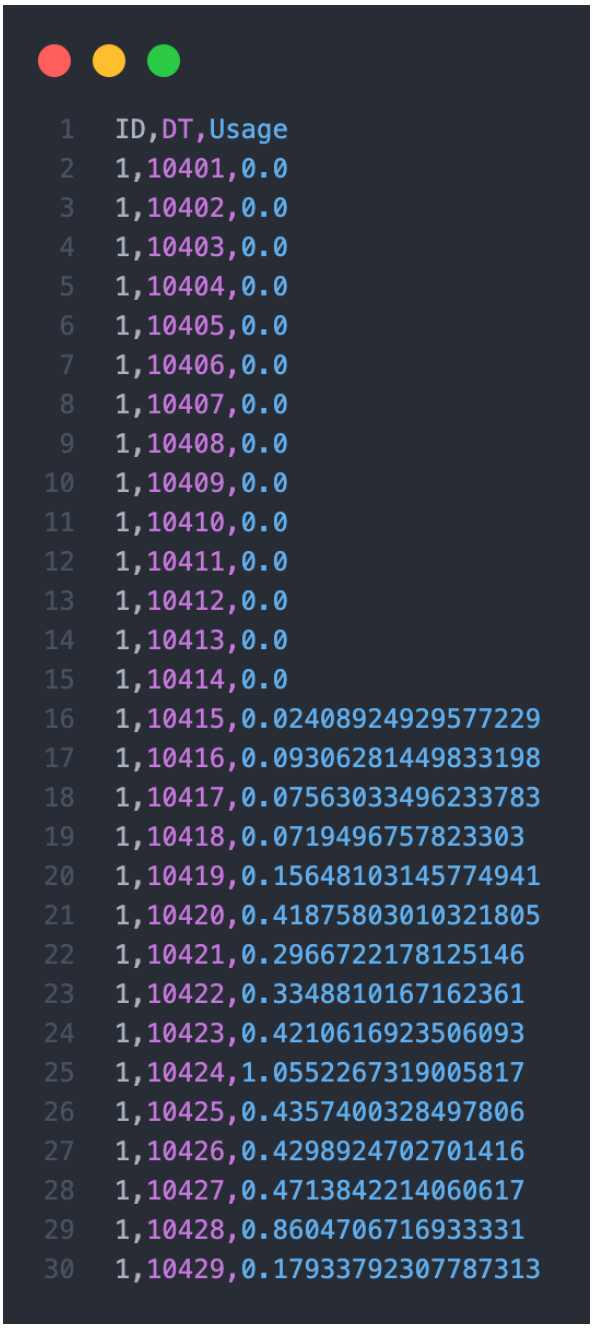
(b) Rating

Figura B.7: Ficheros de *training* de los escenarios de generación *normal* y *rating* para el contador #1



```
ID,DT,Usage
1,10401,0.0
1,10402,0.0
1,10403,0.0
1,10404,0.0
1,10405,0.0
1,10406,0.0
1,10407,0.0
1,10408,0.0
1,10409,0.0
1,10410,0.0
1,10411,0.0
1,10412,0.0
1,10413,0.0
1,10414,0.0
1,10415,0.79
1,10416,0.79
1,10417,0.79
1,10418,0.79
1,10419,0.79
1,10420,0.79
1,10421,0.79
1,10422,0.79
1,10423,0.79
1,10424,0.79
1,10425,0.79
1,10426,0.79
1,10427,0.79
1,10428,0.79
1,10429,0.79
```

(a) Percentile



```
ID,DT,Usage
1,10401,0.0
1,10402,0.0
1,10403,0.0
1,10404,0.0
1,10405,0.0
1,10406,0.0
1,10407,0.0
1,10408,0.0
1,10409,0.0
1,10410,0.0
1,10411,0.0
1,10412,0.0
1,10413,0.0
1,10414,0.0
1,10415,0.02408924929577229
1,10416,0.09306281449833198
1,10417,0.07563033496233783
1,10418,0.0719496757823303
1,10419,0.15648103145774941
1,10420,0.41875803010321805
1,10421,0.2966722178125146
1,10422,0.3348810167162361
1,10423,0.4210616923506093
1,10424,1.0552267319005817
1,10425,0.4357400328497806
1,10426,0.4298924702701416
1,10427,0.4713842214060617
1,10428,0.8604706716933331
1,10429,0.17933792307787313
```

(b) RSA 0.5 3

Figura B.8: Ficheros de *training* de los escenarios de generación *percentile* y *RSA 0.5 3* para el contador #1

Anexos C

Herramientas

NumPy [24]: librería de *Python* que permite crear vectores y matrices multidimensionales, esta ofrece una variedad de funciones para operar de forma rápida con estas matrices.

Pandas [28]: biblioteca de *Python* de código abierto especializada en la manipulación y el análisis de datos.

Matplotlib [27]: biblioteca para crear todo tipo de gráficos en *Python*.

Scikit-learn [26]: librería de aprendizaje automático que admite el aprendizaje supervisado y no supervisado. Proporciona herramientas para el ajuste de modelos, el preprocesamiento de datos, etc.

TensorFlow [25]: biblioteca de código abierto que permite implementar, ejecutar y gestionar algoritmos de *machine learning*.

Keras [29]: librería de código abierto de redes neuronales que funciona como una API que permite acceder a *frameworks* como Tensorflow.

Keras Tuner [14]: marco que simplifica la búsqueda de hiperparámetros. Permite configurar fácilmente un espacio de búsqueda para encontrar los mejores valores de hiperparámetros mediante el uso de ciertos algoritmos de búsqueda disponibles.

Seaborn [33]: biblioteca de visualización de datos de *Python* basada en *matplotlib* que permite crear gráficos estadísticos.

SciPy [30]: colección de algoritmos matemáticos construida en *NumPy*.

Google Colab [16]: permite programar y ejecutar *Pyhton* en el navegador mediante el ofrecimiento de un servicio de *Jupyter* Notebooks [23] alojados.

Beautiful Soup [32]: biblioteca de *Python* para extraer datos de archivos *HTML* y *XML*.

Anexos D

Gestión del proyecto

Este anexo sirve como ampliación del Capítulo 5. Se dedica a representar visualmente la planificación de las distintas tareas a lo largo de la duración del proyecto (Figura D.1), así como el tiempo dedicado al proyecto dividido en categorías (Figuras: D.2, D.3, D.4, D.5 y D.6).

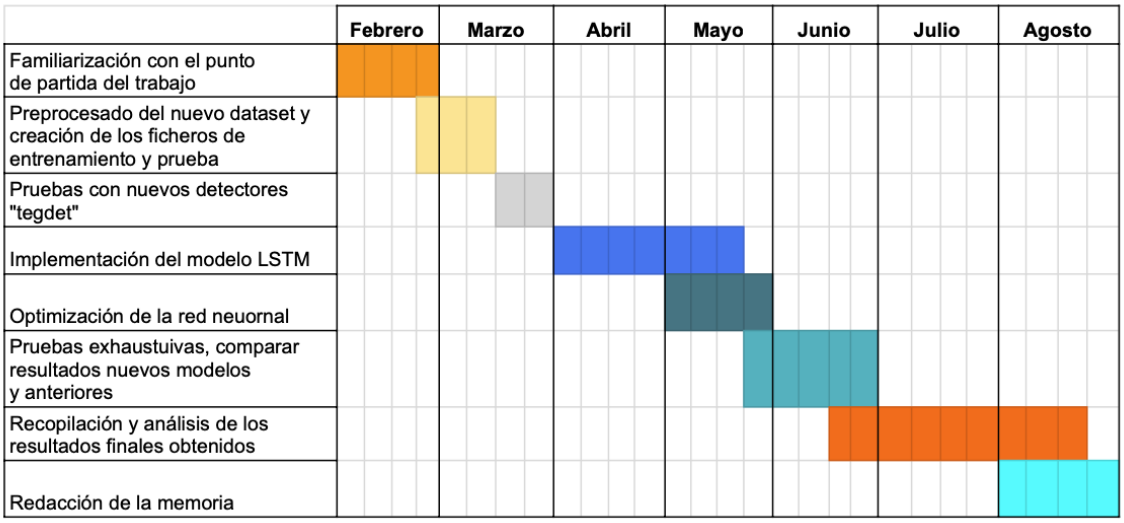


Figura D.1: Diagrama de Gantt del proyecto

Semana	Tarea	Fecha	Horas	Tipo tarea
	Reunión Inicial	22/02/2023	0,5	Reunión
Semana 1	Lectura y estudio del paper	23/02/2023	1,5	Estudio
	Reunión Semanal	01/03/2023	0,5	Reunión
	Revisión estructura del repositorio GitHub	02/03/2023	1	Estudio
Semana 2	Estudio ficheros principales + dudas semanales	07/03/2023	1,5	Gestión
	Reunión Semanal	08/03/2023	0,5	Reunión
	Estudio implementación detectores	09/03/2023	2,5	Estudio
	Correo obtención cluster I3A	10/03/2023	0,5	Gestión
	Tutorial de Tensorflow para Python	13/03/2023	1	Estudio
	Pruebas con la red neuronal implementada	13/03/2023	4	Pruebas
	Lectura paper tegdet	14/03/2023	2	Estudio
Semana 3	Recopilación dudas semanales	14/03/2023	0,5	Gestión
	Reunión semanal	15/03/2023	1	Reunión
	Conexión y familiarización cluster I3A	16/03/2023	3,5	Estudio
	Arreglar errores lanzamiento primeras pruebas en el cluster	16/03/2023	1,5	Implementación
	Lanzamiento pruebas detectores (cluster)	20/03/2023	2	Pruebas
	Cambios en ficheros __init.py__ para pruebas más eficientes	20/03/2023	4	Implementación
	Correo duda Simona	20/03/2023	0,5	Gestión
	Resolver preprocesado con la respuesta del correo	21/03/2023	1	Implementación
	Lanzamiento pruebas detectores (cluster) pt.II	21/03/2023	2	Pruebas
	Lectura segundo paper TegDet	21/03/2023	0,5	Estudio
	Familiarización tegdet	21/03/2023	0,5	Estudio
Semana 4	Lanzamiento pruebas TegDet (cluster)	21/03/2023	2,5	Pruebas
	Reunión semanal	22/03/2023	1	Reunión
	Debugear errores detector PCA-DBSCAN	23/03/2023	4	Implementación
	Lanzamiento pruebas ARIMA(X), PCA-DBSCAN, TegDet	23/03/2023	1	Pruebas
	Preprocesado training dataset + pruebas NN	23/03/2023	2	Pruebas
	Lectura paper Solar	27/03/2023	0,5	Estudio
	Adaptación ficheros detectores Solar	27/03/2023	0,5	Implementación
	Pruebas ataques Solar (Min-Avg, KLD, JSD, NN)	27/03/2023	0,5	Pruebas
	Script parametros NN (capas, neuronas, epochs, act_func)	27/03/2023	1,5	Implementación
	Python Tensorflow for ML	27/03/2023	1	Estudio
	Implementación posibles mejoras NN	27/03/2023	2	Implementación
	Hyperparameter Tuning in Python	28/03/2023	0,5	Estudio
	Script pruebas rendimiento NN (Google Colab)	28/03/2023	5	Implementación
	Overfit y underfit	28/03/2023	0,5	Estudio
Semana 5	Designing NN & Recipe for training NN	28/03/2023	1	Estudio
Semana 6	Reunión semanal	29/03/2023	1	Reunión
	Estudiar scripts generación ataques	09/04/2023	1,5	Análisis
	Análisis de que ataques tiene sentido replicar	09/04/2023	1,5	Análisis
	Estudiar LSTM y CNN para predecir valor consumo	10/04/2023	3	Estudio
	Estudiar paper False Data Injection Attack Detection in Smart Grid	10/04/2023	3	Estudio
	Stock Price Prediction & Forecasting with LSTM	11/04/2023	1	Estudio
	Amazon Stock Forecasting with LSTM	11/04/2023	1	Estudio
Semana 7	Comenzar implementación detector LSTM	11/04/2023	6,5	Implementación

Figura D.2: Fragmento de hoja de cálculo para el seguimiento de esfuerzos

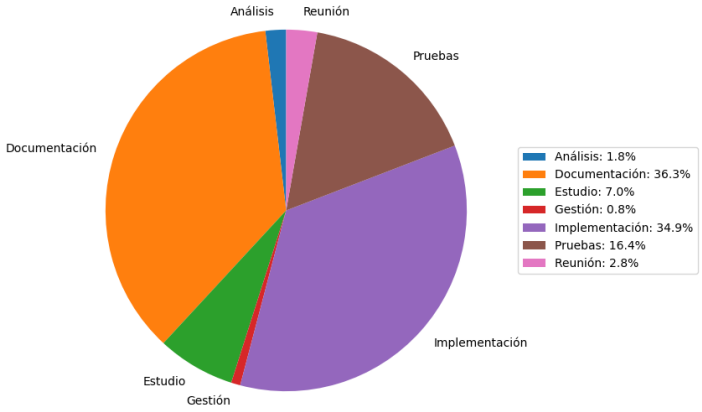


Figura D.3: Porcentaje del tiempo total invertido en cada categoría

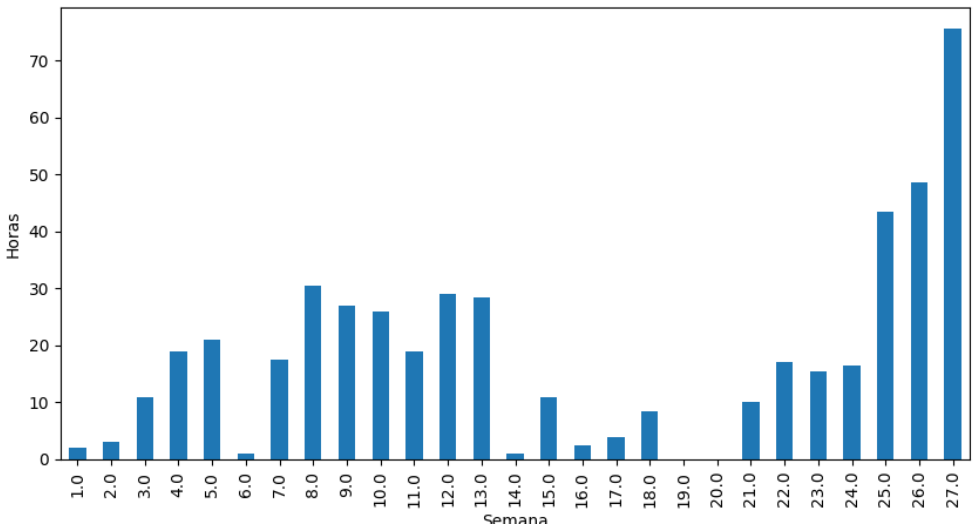


Figura D.4: Distribución del tiempo invertido a lo largo de las semanas

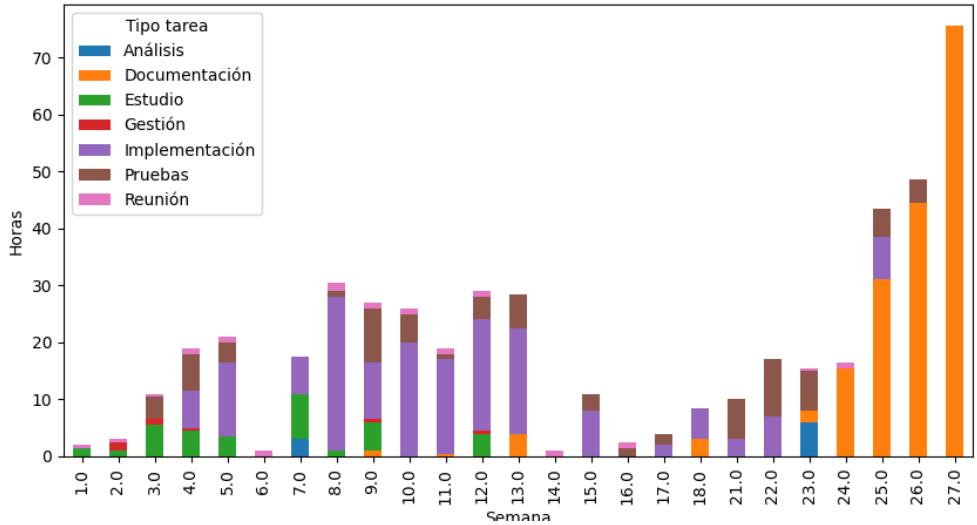


Figura D.5: Distribución del tiempo invertido a cada categoría a lo largo de las semanas

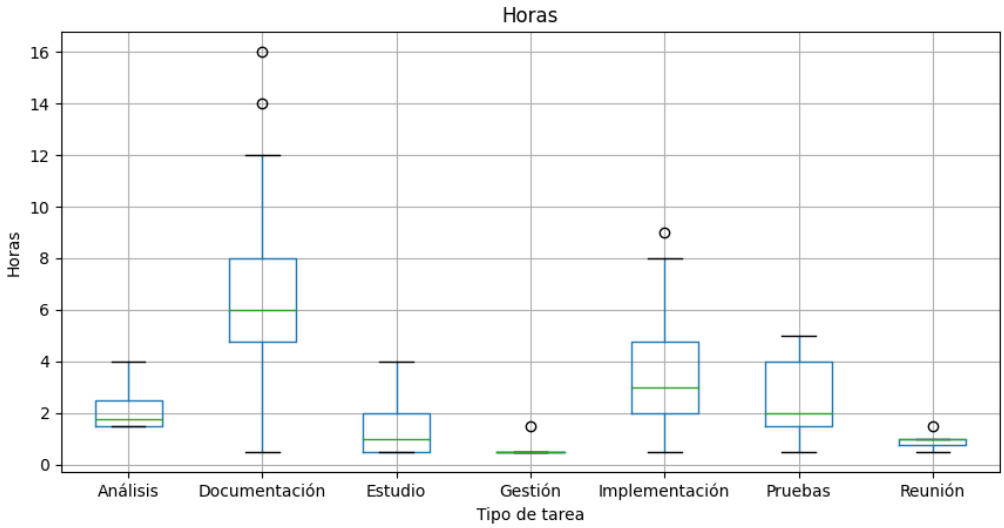


Figura D.6: Gráfico de cajas del tiempo total invertido en cada categoría

Anexos E

Métricas

En este anexo se describen las distintas métricas utilizadas para evaluar y comparar los detectores. Las principales se listan y se explican brevemente a continuación:

- ***Time build***: tiempo que tarda el modelo en construirse.
- ***Time predict***: tiempo que tarda el modelo en las predicciones.
- ***True Positives***: número de veces que el modelo clasifica correctamente que una muestra pertenece a la clase positiva.
- ***True Negatives***: número de veces que el modelo clasifica correctamente que una muestra no pertenece a la clase positiva.
- ***False Positives***: número de veces que el modelo clasifica incorrectamente que una muestra pertenece a la clase positiva cuando en realidad no lo hace.
- ***False Negatives***: número de veces que el modelo clasifica incorrectamente que una muestra no pertenece a la clase positiva cuando en realidad si lo hace.

Además de las métricas principales comentadas anteriormente, también hay otras métricas que se calculan y evalúan, la única distinción con las anteriores es que estas no son obtenidas directamente en el periodo de experimentación, sino que se calculan *a posteriori* con el postprocesamiento de los datos.

- ***Accuracy***: mide con qué frecuencia clasifica correctamente.
- ***Prevalence***: con qué frecuencia ocurre realmente la condición "sí" en la muestra.
- ***Balanced accuracy***: especialmente útil cuando se trabaja con clases desequilibradas, es decir, que una tiene muchos más ejemplos que otra.

- ***Matthews Correlation Coefficient***: métrica que cuantifica la calidad de las predicciones, los valores que pueden dar son -1 si el modelo clasifica al revés, 0 si el rendimiento no mejora el de una predicción aleatoria (p.ej: lanzar una moneda al aire) y 1 si las clasificaciones son perfectas.

- ***Misclass***: representa la proporción de clasificaciones incorrectas respecto al total

- ***True Positive Rate*** (***Recall***): cuando la clase es positiva, con que frecuencia lo predice como tal, es decir, cuando es "sí", con qué frecuencia predice "sí".

- ***True Negative Rate***: cuando la clase es negativa, con que frecuencia lo predice como tal, es decir, cuando es "no", con qué frecuencia predice "no".

- ***False Negative Rate***: cuando la clase es positiva, con que frecuencia lo clasifica en la clase negativa, es decir, cuando es "sí", con qué frecuencia predice "no".

- ***False Positive Rate***: cuando la clase es negativa, con que frecuencia lo clasifica en la clase positiva, es decir, cuando es "no", con qué frecuencia predice "sí".

- ***Precision***: cuando predice "sí", con qué frecuencia es correcto.

- ***Negative Predictive Value***: cuando predice "no" con qué frecuencia es correcto.

- ***F1-score***: se utiliza en lugar de la precisión cuando la muestra tiene una gran cantidad de valores negativos.

time_build	time to build the model	
time_predict	time to predict with the model	
acc	accuracy	$(tp + tn)/(tp + fn + fp + tn)$
tpr (*recall*)	true positive rate	$tp/(tp + fn)$
tnr	true negative rate	$tn/(fp + tn)$
fnr	false negative rate	$fn/(tp + fn)$
fpr	false positive rate	$fp/(tn + fp)$
prev	prevalence	$(tp + fn)/(tp + fn + fp + tn)$
bacc	balanced accuracy	$(tpr + tnr)/2$
mcc	Matthews correlation coefficient	$\frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$
npv	negative predictive value	$tn/(tn + fn)$
$f1score$	f1-score	$2 \cdot \frac{precision \cdot recall}{precision + recall} = 2 \cdot \frac{\frac{tp}{tp+fp} \cdot \frac{tp}{tp+fn}}{\frac{tp}{tp+fp} + \frac{tp}{tp+fn}}$
prev	prevalence	$(tp + fn)/(tp + fn + fp + tn)$

Tabla E.1: Métricas empleadas

Anexos F

Resultados

Este anexo tiene como objetivo presentar los diferentes resultados obtenidos para cada una de las pruebas comentadas en el Capítulo 4.

En primer lugar, se muestran los resultados de los detectores iniciales para cada uno de los conjuntos de datos: electricidad, gas y solar. Seguidamente, se exponen los resultados de la misma evaluación que antes, pero esta vez con los detectores *teg*. A continuación, se dedica la siguiente sección de este anexo a recoger los resultados de los nuevos modelos de detección implementados: *LSTM* y la red neuronal optimizada. Las tablas aportadas en cada uno de los apartados proporcionan información útil sobre la capacidad de detección de cada modelo para cada uno de los diferentes tipos de ataques dependiendo del conjunto de datos evaluado. Finalmente, la última sección se emplea para recopilar distintas pruebas realizadas con la intención de optimizar al máximo estos nuevos modelos de detección.

Cabe destacar que todos los resultados presentados en este anexo se refieren a dos métricas de calidad de los detectores, en concreto *true positive rate* (o *recall*) y *true negative rate*. Las dos métricas proporciona información complementar: *tpr* permite evaluar la capacidad de detectar casos de ataque (datos falsificados) y *tnr* es permite evaluar la capacidad de reconocer los casos normales (datos no falsificados). En ambos casos los valores varían en un intervalo [0,1], cuanto más próximo a 1 es el valor obtenido mejor.

F.1. Detectores iniciales

En esta sección se pueden encontrar las tablas de resultados completas a nivel de escenario de los detectores iniciales (*Min-Avg, ARIMA, ARIMAX, KLD, JSD, NN*). Cada uno de los subapartados se centra en un conjunto de datos, quedando finalmente organizado de la siguiente manera: el primero para la electricidad, el segundo para el gas y el último para la energía solar.

F.1.1. Electricidad (ISSDA CER)

En este apartado, se presentan las Tablas F.1 y F.2 que muestran los resultados anteriores de los detectores iniciales recopilados en [2] y los resultados de la repetición de las pruebas, respectivamente.

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.978	0.934	0.928	0.834	0.867	0.137
tpr	FDI10	0.955	0.002	0.029	0.887	0.909	0.863
	FDI30	0.757	0.002	0.015	0.738	0.789	0.863
	Avg	0.022	0.010	0.028	0.902	0.912	1.000
	RSA_0.25_1.1	0.226	0.030	0.037	0.202	0.203	0.942
	RSA_0.5_3	0.003	0.170	0.185	0.765	0.554	0.952
	Swap	0.022	0.066	0.130	0.166	0.133	0.988

Tabla F.1: Resultados anteriores de los detectores iniciales para el *dataset ISSDA CER* de electricidad

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.983	0.932	0.926	0.838	0.870	0.760
tpr	FDI10	0.950	0.004	0.028	0.884	0.909	0.992
	FDI30	0.729	0.003	0.013	0.736	0.792	0.972
	Avg	0.018	0.010	0.030	0.919	0.931	0.995
	RSA_0.25_1.1	0.212	0.031	0.037	0.224	0.222	0.890
	RSA_0.5_3	0.004	0.173	0.188	0.773	0.550	0.951
	Swap	0.017	0.068	0.131	0.162	0.130	0.953

Tabla F.2: Repetición de los resultados de los detectores iniciales para el *dataset ISSDA CER* de electricidad

A partir de las tablas se pueden sacar las mismas conclusiones que en [2], ya que los resultados son muy similares. Las principales son que el detector *Min-Avg*, es capaz de detectar los ataques *FDI* pero no los demás. Por otro lado, *ARIMA* y *ARIMAX* como era de esperar, tienen un rendimiento similar entre sí, siendo ambos incapaces de reconocer ningún tipo de ataque. *KLD* y *JSD* son buenos detectando ataques *FDI* y *Avg* pero su capacidad de detección baja considerablemente con los ataques *RSA* y

Swap, siendo algo mejor en detectar aquellos ataques *RSA* que reportan el consumo por encima del real.

Por último, como ya se había comentado anteriormente en los apartados en los que se proponía optimizar el modelo de la *NN*, este es excelente detectando todo tipo de ataques, pero debido a su alto número de falsos positivos no es capaz de detectar los escenarios normales. Sin embargo, se puede observar que en la tabla repetida el *tnr* aumenta respecto a la inicial, esto se debe a que después de [2], se introdujo un "sesgo" a la red neuronal para que clasificase con mayor probabilidad los escenarios normales tal y como se comenta en [9] que utiliza esta versión del *NN*. De todas formas, este "sesgo" no es la manera más óptima de mejorar los resultados, por lo que el objetivo es mejorar estos resultados en la red neuronal optimizada sin utilizarlo.

F.1.2. Gas (ISSDA CER)

En este apartado, de igual manera que en el anterior, se encuentran las tablas que muestran los resultados de punto de partida para los detectores iniciales (Tabla F.3) y los resultados de la repetición de la experimentación (Tabla F.4), en este caso trabajando con el conjunto de datos del gas.

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.991	0.934	0.936	0.965	0.971	0.113
tpr	FDI10	0.286	0.000	0.005	0.650	0.807	0.887
	FDI30	0.093	0.000	0.003	0.588	0.661	0.887
	Avg	0.009	0.002	0.007	0.852	0.905	1.000
	RSA_0.25_1.1	0.022	0.028	0.029	0.089	0.047	0.930
	RSA_0.5_3	0.004	0.120	0.122	0.355	0.049	0.933
	Swap	0.009	0.066	0.125	0.035	0.029	0.991

Tabla F.3: Resultados anteriores de los detectores iniciales para el *dataset ISSDA CER* de gas

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.993	0.936	0.937	0.968	0.975	0.669
tpr	FDI10	0.298	0.000	0.005	0.650	0.801	0.908
	FDI30	0.098	0.000	0.003	0.589	0.667	0.895
	Avg	0.007	0.001	0.006	0.855	0.903	0.977
	RSA_0.25_1.1	0.022	0.027	0.028	0.087	0.044	0.793
	RSA_0.5_3	0.003	0.120	0.123	0.367	0.041	0.855
	Swap	0.007	0.064	0.126	0.032	0.025	0.890

Tabla F.4: Repetición de los resultados de los detectores iniciales para el *dataset ISSDA CER* de gas

Los resultados recogidos en las tablas, confirman lo que se ha expuesto con el

conjunto de datos de la electricidad. Aunque, en general, los resultados para el gas empeoran (menos en los escenarios normales) todos los detectores siguen la misma tendencia, a excepción de *Min-Avg* que con estos datos pierde su capacidad para detectar ataques *FDI*.

F.1.3. Solar (Ausgrid)

En este apartado se recogen los resultados obtenidos al evaluar los detectores iniciales con el nuevo conjunto de datos. Por un lado, se presenta la Tabla F.5 en la que se prueban los escenarios de consumo, al igual que se ha hecho en los anteriores apartados. Mientras que, la Tabla F.6 expone los resultados de los nuevos escenarios.

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.940	0.940	0.934	0.858	0.873	0.780
tpr	FDI10	0.967	0.004	0.045	0.944	0.951	0.999
	FDI30	0.816	0.004	0.025	0.817	0.843	0.999
	Avg	0.061	0.006	0.032	0.836	0.870	0.999
	RSA_0.25_1.1	0.361	0.028	0.037	0.404	0.393	0.970
	RSA_0.5_3	0.006	0.155	0.178	0.611	0.363	0.978
	Swap	0.060	0.060	0.109	0.142	0.127	0.907

Tabla F.5: Repetición de los resultados de los detectores iniciales para el *dataset Solar Ausgrid* y los escenarios de consumo

Métrica	**Escenario**	**Min-Avg**	**ARIMA**	**ARIMAX**	**KLD**	**JSD**	**NN**
tnr	Normal	0.971	0.907	0.907	0.914	0.907	0.897
tpr	Percentile	0.012	0.372	0.320	0.999	0.999	0.894
	Rating	0.000	0.493	0.499	0.963	0.963	0.907
	RSA_0.5_3	0.013	0.168	0.197	0.228	0.095	0.814

Tabla F.6: Repetición de los resultados de los detectores iniciales para el *dataset Solar Ausgrid* y los escenarios de generación

En resumen, las conclusiones que se pueden extraer respecto a los resultados obtenidos para los escenarios de consumo serían las mismas que para el conjunto de datos de la electricidad, ya que los resultados son muy similares y todos los detectores siguen la misma tendencia.

En lo que respecta a los escenarios de generación, se puede observar que todos los modelos son muy buenos detectando los escenarios normales pero muy malos para los escenarios de ataque. Concretamente, el detector *Min-Avg* es incapaz de detectar ningún ataque, esto es debido a que se basa en etiquetar como anómalos aquellos casos donde el valor es inferior al mínimo del promedio de las semanas de entrenamiento y al tratarse de ataques que buscan reportar un valor por encima del real tiene sentido

que no los reconozca como tal. Por otro lado, *ARIMA* y *ARIMAX* aunque son algo mejores, ninguno consigue detectar más de la mitad de las veces. En cuanto a *KLD* y *JSD*, son muy buenos detectando ataques *Percentile* y *Rating*, pero su rendimiento cae en picado al tratar de detectar el ataque *RSA*. Por último, la red neuronal sigue la misma tendencia que antes.

F.2. Detectores Teg

En esta sección se pueden encontrar las tablas de resultados completas a nivel de escenario de los 28 detectores *teg* que se han evaluado. Esta sección se divide igual que la anterior de los detectores iniciales, cada subapartado se centra en uno de los conjuntos de datos utilizados.

F.2.1. Electricidad (ISSDA CER)

Este apartado recoge los resultados de los 28 detectores utilizando el conjunto de datos *ISSDA CER* Electricidad. Se dividen en distintas tablas para mantenerlos ordenados y visibles para su posterior análisis.

En conclusión, los resultados obtenidos indican detectores variopintos en cuanto a la detección de los distintos escenarios, pese a que ninguno es tan bueno detectando todos los tipos de ataques como el modelo *NN*, entre ellos se puede destacar algunos que sí presentan cierto grado de consistencia, por ejemplo: *JS*, *Bhattacharyya*, *Squaredchord* y *Clark* entre otros. Por otro lado, está el detector *Lorentzian* que es muy bueno detectando escenarios normales pero incapaz de detectar escenarios de ataque.

Métrica	**Escenario**	**Hamming**	**Cosine**	**Jaccard**	**Dice**	**KL**	**Jeffreys**	**JS**
tnr	Normal	0.930	0.856	0.924	0.925	0.325	0.887	0.844
tpr	FDI10	0.964	0.753	0.918	0.918	0.860	0.947	0.912
	FDI30	0.889	0.571	0.725	0.726	0.655	0.829	0.732
	Avg	0.878	0.900	0.887	0.887	0.939	0.936	0.946
	RSA_0.25_1.1	0.334	0.262	0.214	0.212	0.656	0.386	0.359
	RSA_0.5_3	0.099	0.638	0.064	0.028	0.983	0.533	0.822
	Swap	0.369	0.201	0.092	0.091	0.795	0.540	0.496

Tabla F.7: Tabla del primer grupo de detectores *teg* para el *dataset ISSDA CER* de electricidad

Métrica	Escenario	Euclidean	Cityblock	Chebyshev	Minkowski	Braycurtis
tnr	Normal	0.996	0.852	0.896	0.890	0.852
tpr	FDI10	0.919	0.892	0.914	0.916	0.892
	FDI30	0.744	0.679	0.722	0.736	0.679
	Avg	0.866	0.937	0.764	0.814	0.937
	RSA_0.25_1.1	0.275	0.319	0.232	0.249	0.319
	RSA_0.5_3	0.555	0.798	0.538	0.542	0.798
	Swap	0.197	0.353	0.146	0.164	0.353

Tabla F.8: Tabla del segundo grupo de detectores *teg* para el *dataset ISSDA CER* de electricidad

Métrica	Escenario	Gower	Soergel	Kulczynski	Canberra	Lorentzian	Bhattacharyya
tnr	Normal	0.852	0.851	0.852	0.860	0.950	0.847
tpr	FDI10	0.892	0.892	0.892	0.971	0.000	0.925
	FDI30	0.679	0.679	0.679	0.894	0.001	0.760
	Avg	0.937	0.937	0.937	0.936	0.001	0.950
	RSA_0.25_1.1	0.319	0.319	0.319	0.424	0.017	0.377
	RSA_0.5_3	0.798	0.798	0.798	0.690	0.005	0.818
	Swap	0.353	0.353	0.353	0.665	0.009	0.529

Tabla F.9: Tabla del tercer grupo de detectores *teg* para el *dataset ISSDA CER* de electricidad

Métrica	Escenario	Hellinger	Matusita	Squaredchord	Pearson	Neyman	Squared
tnr	Normal	0.847	0.847	0.847	0.337	0.931	0.844
tpr	FDI10	0.925	0.925	0.925	0.589	0.876	0.902
	FDI30	0.760	0.760	0.760	0.471	0.572	0.711
	Avg	0.950	0.950	0.950	0.832	0.857	0.944
	RSA_0.25_1.1	0.378	0.378	0.377	0.633	0.128	0.347
	RSA_0.5_3	0.818	0.818	0.818	0.981	0.087	0.821
	Swap	0.530	0.530	0.529	0.753	0.322	0.464

Tabla F.10: Tabla del cuarto grupo de detectores *teg* para el *dataset ISSDA CER* de electricidad

Métrica	Escenario	Probsymmetric	Divergence	Clark	Additivesymmetric
tnr	Normal	0.844	0.890	0.890	0.854
tpr	FDI10	0.902	0.970	0.970	0.411
	FDI30	0.711	0.896	0.896	0.275
	Avg	0.944	0.930	0.930	0.831
	RSA_0.25_1.1	0.347	0.389	0.389	0.158
	RSA_0.5_3	0.821	0.463	0.463	0.813
	Swap	0.464	0.605	0.605	0.307

Tabla F.11: Tabla del último grupo de detectores *teg* para el *dataset ISSDA CER* de electricidad

F.2.2. Gas (ISSDA CER)

Este apartado se encarga de hacer lo mismo que el anterior, es decir, ilustrar las diferentes tablas que guardan los resultados de los 28 detectores que han sido evaluados, en este caso con los datos del gas. Por lo general, estos resultados siguen la tendencia de perdida de eficacia en la detección de ataques y mejora en la detección de escenarios normales.

Métrica	**Escenario**	**Hamming**	**Cosine**	**Jaccard**	**Dice**	**KL**	**Jeffreys**	**JS**
tnr	Normal	0.974	0.985	0.968	0.966	0.330	0.909	0.970
tpr	FDI10	0.639	0.160	0.249	0.265	0.405	0.841	0.329
	FDI30	0.342	0.131	0.098	0.105	0.673	0.731	0.388
	Avg	0.597	0.802	0.805	0.843	0.811	0.923	0.818
	RSA_0.25_1.1	0.056	0.010	0.045	0.047	0.814	0.291	0.163
	RSA_0.5_3	0.472	0.047	0.029	0.029	0.955	0.816	0.431
	Swap	0.031	0.019	0.035	0.038	0.696	0.146	0.055

Tabla F.12: Tabla del primer grupo de detectores *teg* para el *dataset ISSDA CER* de gas

Métrica	**Escenario**	**Euclidean**	**Cityblock**	**Chebyshev**	**Minkowski**	**Braycurtis**
tnr	Normal	0.981	0.989	0.981	0.981	0.989
tpr	FDI10	0.173	0.046	0.177	0.177	0.046
	FDI30	0.058	0.058	0.057	0.057	0.058
	Avg	0.841	0.797	0.824	0.837	0.797
	RSA_0.25_1.1	0.020	0.020	0.024	0.023	0.020
	RSA_0.5_3	0.028	0.073	0.030	0.028	0.073
	Swap	0.025	0.022	0.025	0.025	0.022

Tabla F.13: Tabla del segundo grupo de detectores *teg* para el *dataset ISSDA CER* de gas

Métrica	**Escenario**	**Gower**	**Soergel**	**Kulczynski**	**Canberra**	**Lorentzian**	**Bhattacharyya**
tnr	Normal	0.989	0.989	0.989	0.931	0.904	0.956
tpr	FDI10	0.046	0.046	0.046	0.841	0.001	0.478
	FDI30	0.058	0.058	0.058	0.664	0.006	0.508
	Avg	0.797	0.797	0.797	0.901	0.000	0.832
	RSA_0.25_1.1	0.020	0.020	0.020	0.274	0.109	0.242
	RSA_0.5_3	0.073	0.073	0.073	0.844	0.070	0.587
	Swap	0.022	0.022	0.022	0.106	0.077	0.083

Tabla F.14: Tabla del tercer grupo de detectores *teg* para el *dataset ISSDA CER* de gas

Métrica	Escenario	Hellinger	Matusita	Squaredchord	Pearson	Neyman	Squared
tnr	Normal	0.956	0.956	0.956	0.320	0.952	0.978
tpr	FDI10	0.479	0.478	0.478	0.471	0.520	0.222
	FDI30	0.509	0.509	0.508	0.722	0.282	0.289
	Avg	0.832	0.832	0.832	0.810	0.783	0.811
	RSA_0.25_1.1	0.242	0.242	0.242	0.821	0.057	0.104
	RSA_0.5_3	0.587	0.587	0.587	0.955	0.129	0.309
	Swap	0.083	0.083	0.083	0.709	0.096	0.040

Tabla F.15: Tabla del cuarto grupo de detectores *teg* para el *dataset ISSDA CER* de gas

Métrica	Escenario	Probsymmetric	Divergence	Clark	Additivesymmetric
tnr	Normal	0.946	0.946	0.923	0.923
tpr	FDI10	0.821	0.822	0.291	0.244
	FDI30	0.611	0.611	0.462	0.260
	Avg	0.890	0.892	0.788	0.788
	RSA_0.25_1.1	0.190	0.190	0.244	0.122
	RSA_0.5_3	0.777	0.777	0.260	0.291
	Swap	0.076	0.076	0.122	0.462

Tabla F.16: Tabla del último grupo de detectores *teg* para el *dataset ISSDA CER* de gas

F.2.3. Solar (Ausgrid)

En este apartado, de igual forma que en los dos anteriores, se presentan los resultados de los 28 detectores. En este caso han sido evaluados utilizando el conjunto de datos *Solar Ausgrid*.

Escenarios de consumo

Las conclusiones extraídas tras la evaluación de los escenarios de consumo es similar a lo que se había visto con los detectores iniciales. Es decir, que estos tienen una precisión muy similar a cuando son probados con el conjunto de datos *ISSDA CER* Electricidad. A continuación, se muestran el listado completo de las tablas de resultados.

Métrica	Escenario	Hamming	Cosine	Jaccard	Dice	KL	Jeffreys	JS
tnr	Normal	0.901	0.881	0.870	0.871	0.385	0.854	0.863
tpr	FDI10	0.953	0.788	0.958	0.958	0.869	0.949	0.940
	FDI30	0.850	0.675	0.852	0.851	0.686	0.807	0.778
	Avg	0.858	0.808	0.771	0.776	0.872	0.907	0.905
	RSA_0.25_1.1	0.272	0.379	0.397	0.396	0.721	0.405	0.438
	RSA_0.5_3	0.115	0.515	0.022	0.015	0.987	0.504	0.723
	Swap	0.240	0.134	0.158	0.157	0.760	0.409	0.277

Tabla F.17: Tabla del primer grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de consumo

Métrica	Escenario	Euclidean	Cityblock	Chebyshev	Minkowski	Braycurtis
tnr	Normal	0.865	0.873	0.869	0.865	0.873
tpr	FDI10	0.956	0.938	0.956	0.956	0.938
	FDI30	0.840	0.767	0.840	0.840	0.767
	Avg	0.733	0.887	0.553	0.651	0.887
	RSA_0.25_1.1	0.426	0.441	0.341	0.386	0.441
	RSA_0.5_3	0.408	0.700	0.400	0.401	0.700
	Swap	0.177	0.217	0.152	0.162	0.217

Tabla F.18: Tabla del segundo grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de consumo

Métrica	Escenario	Gower	Soergel	Kulczynski	Canberra	Lorentzian	Bhattacharyya
tnr	Normal	0.873	0.873	0.873	0.822	0.946	0.860
tpr	FDI10	0.938	0.938	0.937	0.959	0.000	0.943
	FDI30	0.767	0.768	0.764	0.870	0.006	0.793
	Avg	0.887	0.887	0.883	0.902	0.000	0.910
	RSA_0.25_1.1	0.441	0.442	0.440	0.508	0.019	0.445
	RSA_0.5_3	0.700	0.702	0.695	0.703	0.004	0.713
	Swap Swap	0.217	0.217	0.216	0.527	0.018	0.306

Tabla F.19: Tabla del tercer grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de consumo

Métrica	Escenario	Hellinger	Matusita	Squaredchord	Pearson	Neyman	Squared
tnr	Normal	0.860	0.860	0.860	0.398	0.897	0.867
tpr	FDI10	0.943	0.943	0.943	0.614	0.868	0.936
	FDI30	0.795	0.795	0.794	0.525	0.592	0.765
	Avg	0.913	0.913	0.912	0.624	0.840	0.896
	RSA_0.25_1.1	0.446	0.446	0.445	0.690	0.227	0.435
	RSA_0.5_3	0.716	0.716	0.714	0.984	0.131	0.721
	Swap	0.307	0.307	0.306	0.746	0.261	0.256

Tabla F.20: Tabla del cuarto grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de consumo

Métrica	**Escenario**	**Probsymmetric**	**Divergence**	**Clark**	**Additivesymmetric**
tnr	Normal	0.867	0.853	0.852	0.882
tpr	FDI10	0.936	0.959	0.959	0.634
	FDI30	0.765	0.867	0.867	0.556
	Avg	0.896	0.897	0.897	0.622
	RSA_0.25_1.1	0.435	0.448	0.449	0.350
	RSA_0.5_3	0.721	0.517	0.517	0.689
	Swap	0.256	0.451	0.451	0.205

Tabla F.21: Tabla del último grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de consumo

Escenarios de generación

Tras analizar las tablas que se presentan a continuación, se puede concluir que, sorprendentemente, los resultados encontrados difieren mucho respecto a cuando se han probado los detectores iniciales con estos mismos datos y para los mismos escenarios. En este caso, se puede observar como la gran mayoría de detectores, a exclusión de algunos como *Lorentzian*, son excelentes detectando tanto los escenarios normales como los distintos escenarios de ataque.

Métrica	**Escenario**	**Hamming**	**Cosine**	**Jaccard**	**Dice**	**KL**	**Jeffreys**	**JS**
tnr	Normal	0.884	0.927	0.907	0.907	0.228	0.849	0.850
tpr	Percentile	0.993	1.000	0.976	0.976	0.996	1.000	1.000
	Rating	0.993	1.000	0.976	0.976	1.000	1.000	1.000
	RSA_0.5_3	0.882	0.837	0.035	0.035	0.987	0.989	0.984

Tabla F.22: Tabla del primer grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de generación

Métrica	**Escenario**	**Euclidean**	**Cityblock**	**Chebyshev**	**Minkowski**	**Braycurtis**
tnr	Normal	0.897	0.898	0.902	0.900	0.898
tpr	Percentile	0.999	1.000	0.999	0.999	1.000
	Rating	1.000	1.000	0.999	0.999	1.000
	RSA_0.5_3	0.338	0.836	0.225	0.279	0.836

Tabla F.23: Tabla del segundo grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de generación

Métrica	**Escenario**	**Gower**	**Soergel**	**Kulczynski**	**Canberra**	**Lorentzian**	**Bhattacharyya**
tnr	Normal	0.898	0.898	0.898	0.857	0.942	0.843
tpr	Percentile	1.000	1.000	1.000	1.000	0.000	1.000
	Rating	1.000	1.000	1.000	1.000	0.000	1.000
	RSA_0.5_3	0.827	0.836	0.836	0.990	0.001	0.990

Tabla F.24: Tabla del tercer grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de generación

Métrica	Escenario	Hellinger	Matusita	Squaredchord	Pearson	Neyman	Squared
tnr	Normal	0.843	0.843	0.843	0.237	0.900	0.859
tpr	Percentile	1.000	1.000	1.000	0.996	0.964	1.000
	Rating	1.000	1.000	1.000	1.000	0.993	1.000
	RSA_0.5_3	0.990	0.990	0.990	0.987	0.599	0.975

Tabla F.25: Tabla del cuarto grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de generación

Métrica	Escenario	Probsymmetric	Divergence	Clark	Additivesymmetric
tnr	Normal	0.859	0.863	0.863	0.864
tpr	Percentile	1.000	1.000	1.000	0.983
	Rating	1.000	1.000	1.000	0.163
	RSA_0.5_3	0.975	0.977	0.977	0.573

Tabla F.26: Tabla del último grupo de detectores *teg* para el *dataset Solar Ausgrid* y escenarios de generación

F.3. Red neuronal optimizada y *LSTM*

En este apartado se recogen los resultados de los nuevos modelos de detección implementados. En las tablas aportadas se observa la capacidad de ambos modelos (*NN_v2* y *LSTM*) para detectar cada uno de los escenarios.

F.3.1. Electricidad (ISSDA CER)

En la Tabla F.27 se pueden observar los resultados de los nuevos modelos sobre el conjunto de datos *ISSDA CER* Electricidad. A partir de esta, se concluye que el modelo basado en redes neuronales es casi excelente detectando todos los escenarios para los que ha sido entrenado. Por otro lado, el modelo basado en redes neuronales recurrentes consigue acercarse a los resultados del modelo *NN_v2* llegando incluso a superarle en los ataques *FDI*. Para el resto de ataques la precisión es similar, menos en *RSA 0.5 3* que *LSTM* solo es capaz de clasificarlo correctamente un 55 % del total de veces.

Métrica	Escenario	NN_v2	LSTM
tnr	Normal	0.838	0.762
tpr	FDI10	0.961	0.995
	FDI30	0.929	0.944
	Avg	0.988	0.829
	RSA_0.25_1.1	0.828	0.777
	RSA_0.5_3	0.952	0.510
	Swap	0.971	0.796

Tabla F.27: Resultados de *NN_v2* y *LSTM* para el *dataset ISSDA CER* de electricidad

F.3.2. Gas (ISSDA CER)

Estos detectores, como era de esperar, también en términos generales presenta peores resultados bajo este conjunto de datos. El modelo *NN_v2* se ve afectado principalmente en el escenario normal y en *RSA 0.25 1.1*, el resto de escenarios no tienen una pérdida significativa respecto al *dataset* de la electricidad. Sin embargo, el modelo *LSTM* sí que se ve bastante penalizado en todos los escenarios, menos en los *FDI* que sigue obteniendo una precisión considerablemente alta.

Métrica	**Escenario**	**NN_v2**	**LSTM**
tnr	Normal	0.693	0.542
tpr	FDI10	0.912	0.959
	FDI30	0.855	0.845
	Avg	0.976	0.560
	RSA_0.25_1.1	0.750	0.593
	RSA_0.5_3	0.909	0.304
	Swap	0.954	0.532

Tabla F.28: Resultados de *NN_v2* y *LSTM* para el *dataset ISSDA CER* de gas

F.3.3. Solar (Ausgrid)

En este apartado se pueden encontrar los resultados de los modelos al ser evaluados con el conjunto de datos Solar tanto para los escenarios de consumo (Tabla F.29) como para los de generación (Tabla F.30). Los resultados obtenidos para los escenarios de consumo, tal y como se había visto anteriormente con el resto de detectores, estos son muy parecidos a los que se obtienen utilizando el *ISSDA CER* Electricidad. Finalmente, basándose en lo observado con los otros modelos evaluados, se esperaba una mejora general de los resultados y esto efectivamente ha sido así, llegando a conseguir unas precisiones casi perfectas tanto con *NN_v2* como con *LSTM*.

Métrica	**Escenario**	**NN_v2**	**LSTM**
tnr	Normal	0.763	0.852
tpr	FDI10	0.985	1.000
	FDI30	0.938	0.979
	Avg	0.994	0.955
	RSA_0.25_1.1	0.895	0.873
	RSA_0.5_3	0.939	0.590
	Swap	0.929	0.847

Tabla F.29: Resultados de *NN_v2* y *LSTM* para el *dataset Solar Ausgrid* y los escenarios de consumo

Métrica	Escenario	NN_v2	LSTM
tnr	Normal	0.930	0.984
tpr	Percentile	0.927	0.987
	Rating	0.926	0.983
	RSA_0.5_3	0.928	0.878

Tabla F.30: Resultados de *NN_v2* y *LSTM* para el *dataset Solar Ausgrid* y los escenarios de generación

F.4. Otras pruebas

F.4.1. Red neuronal optimizada

Este apartado recoge los resultados de dos pruebas que se desarrollaron con la intención de maximizar la precisión del modelo *NN_v2*. Estas son, por un lado, pruebas para conocer el valor óptimo del tamaño de la ventana y, por otro lado, pruebas para el proceso de optimización de hiperparámetros, donde se muestra el formato del fichero que guarda los resultados del proceso explicado en la Sección 3.4.2.

Pruebas tamaño de ventana

En esta sección se muestra la precisión para cada tipo de escenario dependiendo del conjunto de datos. En la Sección 3.4.1 ya se ilustraron las métricas resumidas y a partir de estas se concluyó que el tamaño óptimo era una ventana de 7 días porque encontraba un punto medio entre realizar un número significativo de predicciones y una mejor precisión del modelo.

Métrica	Escenario	NN_v2_2	NN_v2_3	NN_v2_5	NN_v2_7	NN_v2_10	NN_v2_14	NN_v2_30
tnr	Normal	0.835	0.871	0.857	0.869	0.891	0.910	0.921
tpr	FDI10	0.944	0.949	0.965	0.973	0.982	0.990	1.000
	FDI30	0.918	0.945	0.926	0.945	0.963	0.967	0.976
	Avg	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	RSA_0.25_1.1	0.809	0.814	0.829	0.846	0.832	0.848	0.904
	RSA_0.5_3	0.940	0.961	0.961	0.967	0.976	0.974	0.976
	Swap	0.987	0.990	1.000	1.000	1.000	0.992	1.000

Tabla F.31: Resultados de *NN_v2* con distintos tamaños de ventana utilizando el *dataset ISSDA CER* Electricidad

Métrica	Escenario	NN_v2_2	NN_v2_3	NN_v2_5	NN_v2_7	NN_v2_10	NN_v2_14	NN_v2_30
tnr	Normal	0.733	0.797	0.802	0.815	0.803	0.804	0.753
tpr	FDI10	0.918	0.896	0.922	0.926	0.935	0.940	0.941
	FDI30	0.842	0.859	0.865	0.892	0.886	0.900	0.903
	Avg	0.988	0.990	0.994	0.997	0.996	0.995	1.000
	RSA_0.25_1.1	0.751	0.731	0.715	0.734	0.722	0.757	0.792
	RSA_0.5_3	0.912	0.922	0.948	0.949	0.973	0.985	0.991
	Swap	0.940	0.951	0.988	0.978	0.988	0.997	1.000

Tabla F.32: Resultados de *NN_v2* con distintos tamaños de ventana utilizando el *dataset ISSDA CER* Gas

Métrica	Escenario	NN_v2_2	NN_v2_3	NN_v2_5	NN_v2_7	NN_v2_10	NN_v2_14	NN_v2_30
tnr	Normal	0.738	0.739	0.723	0.692	0.662	0.662	0.667
tpr	FDI10	0.994	0.998	0.994	0.998	0.995	1.000	0.998
	FDI30	0.938	0.947	0.959	0.954	0.953	0.947	0.947
	Avg	0.997	1.000	0.999	0.999	0.999	0.999	1.000
	RSA_0.25_1.1	0.912	0.926	0.927	0.933	0.932	0.941	0.941
	RSA_0.5_3	0.913	0.923	0.936	0.936	0.945	0.949	0.949
	Swap	0.809	0.842	0.811	0.822	0.826	0.815	0.856

Tabla F.33: Resultados de *NN_v2* con distintos tamaños de ventana utilizando el *dataset Solar Ausgrid* para escenarios de consumo

Métrica	Escenario	NN_v2_2	NN_v2_3	NN_v2_5	NN_v2_7	NN_v2_10	NN_v2_14	NN_v2_30
tnr	Normal	0.979	0.986	0.995	0.994	0.998	0.998	0.999
tpr	Percentile	0.900	0.899	0.900	0.900	0.898	0.899	0.900
	Rating	0.900	0.900	0.900	0.900	0.900	0.900	0.900
	RSA_0.5_3	0.859	0.875	0.892	0.897	0.898	0.899	0.900

Tabla F.34: Resultados de *NN_v2* con distintos tamaños de ventana utilizando el *dataset Solar Ausgrid* para escenarios de generación

Optimización de hiperparámetros

Esta sección sirve como ampliación de la Sección 3.4.2 ya que se utiliza únicamente para mostrar el formato del fichero que recoge los resultados del *tuner* de hiperparámetros. Tal y como se puede observar en la Figura F.1, el fichero guarda la mejor combinación para cada evaluación, por tanto, al final del proceso se tienen tantas combinaciones óptimas como evaluaciones específicas que se le hayan hecho al modelo.

Para entender esto bien basta con mirar el fichero, la primera evaluación que se le hace a la red neuronal es para el contador con identificador #256 del *dataset* solar. Para este contador concreto, el algoritmo ha concluido que los valores que resultaban en una mejor precisión eran los que se recogen en la segunda línea.

Nota: se ha evaluado únicamente el escenario normal, no se han considerado los escenarios de ataque por motivos del tiempo y por los recursos computacionales que esto requeriría.

```
NN (meter_id = 256, window = 30, dataset = solar)
{'n_layers': 1, 'batch_size': 232, 'neuron_units': 256, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 281, window = 30, dataset = solar)
{'n_layers': 3, 'batch_size': 8, 'neuron_units': 384, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 2049, window = 30, dataset = gas)
{'n_layers': 2, 'batch_size': 8, 'neuron_units': 128, 'tuner/epochs': 4, 'tuner/initial_epoch': 0, 'tuner/bracket': 1, 'tuner/round': 0}
=============================================
NN (meter_id = 6977, window = 30, dataset = electricity)
{'n_layers': 1, 'batch_size': 136, 'neuron_units': 32, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 167, window = 30, dataset = solar)
{'n_layers': 2, 'batch_size': 8, 'neuron_units': 32, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 40, window = 30, dataset = solar)
{'n_layers': 4, 'batch_size': 72, 'neuron_units': 224, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 239, window = 30, dataset = solar)
{'n_layers': 1, 'batch_size': 8, 'neuron_units': 224, 'tuner/epochs': 10, 'tuner/initial_epoch': 4, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0014'}
=============================================
NN (meter_id = 2223, window = 30, dataset = gas)
{'n_layers': 4, 'batch_size': 88, 'neuron_units': 448, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 4613, window = 30, dataset = electricity)
{'n_layers': 2, 'batch_size': 56, 'neuron_units': 192, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 165, window = 30, dataset = solar)
{'n_layers': 3, 'batch_size': 184, 'neuron_units': 32, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
NN (meter_id = 196, window = 30, dataset = solar)
{'n_layers': 3, 'batch_size': 24, 'neuron_units': 160, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 2, 'tuner/round': 0}
=============================================
```

Figura F.1: Fichero que recoge los resultados del proceso de optimización de hiperparámetros de la *NN_v2*

F.4.2. LSTM (Long Short-Term Memory)

En este apartado se especifican las mismas pruebas que en el apartado anterior, pero en este caso para el modelo *LSTM*. Además, se añade una sección que analiza los resultados de distintas versiones del mismo modelo.

Versiones del modelo

En esta sección se van a distinguir tres versiones del modelo *LSTM*: v1, v2 y v3. Cada una de estas versiones se distingue únicamente por los *features* o calificadores estadísticos que se recogen en los datos de cada muestra para intentar aportar mayor información y por consiguiente mejorar la precisión del modelo.

Primeramente, se definen los *features* utilizados en la primera versión que solo aportan información a nivel de día.

- Media de los valores registrados ese día
- Moda de los valores registrados ese día
- Valor máximo del día
- Valor mínimo del día
- Desviación típica del día
- Coeficiente de variación

De otro modo, los calificadores de la segunda versión solo informan al nivel de la media hora de la muestra actual.

- Valor del día anterior en la misma media hora
- Valor del día anterior en la siguiente media hora
- Valor previo registrado
- Media de los valores registrados hasta el momento en la misma media hora que la actual
- Valor actual - Valor previo
- Valor máximo de los valores registrados hasta el momento en la misma media hora que la actual
- Valor mínimo de los valores registrados hasta el momento en la misma media hora que la actual

En último lugar, la tercera versión o la versión definitiva cuenta con todos los anteriores pese a no haber conseguido resultados muy diferenciadores respecto a las otras.

De manera aclaratoria, se explica la experimentación llevada a cabo para la obtención de los resultados. Esta ha consistido en seleccionar 20 contadores aleatoriamente para cada uno de los conjuntos de datos y evaluarlos para las tres versiones del modelo. Por tanto, se evalúan 20 contadores del conjunto de la electricidad, del gas, y de solar. Los contadores del solar se evalúan dos veces, una para los escenarios de generación y otra para los de consumo. A continuación, se listan los ficheros donde se encuentran los resultados [22] y seguidamente se muestran las tablas que los representan visualmente.

```
/script_results/electricity_LSTM_versions.csv
/script_results/gas_LSTM_versions.csv
/script_results/solar_consumption_LSTM_versions.csv
/script_results/solar_generation_LSTM_versions.csv
```

Métrica	Escenario	LSTM_v1	LSTM_v2	LSTM_v3
tnr	Normal	0.707	0.739	0.792
tpr	FDI10	0.995	0.995	0.991
	FDI30	0.934	0.930	0.907
	Avg	0.783	0.725	0.792
	RSA_0.25_1.1	0.739	0.773	0.750
	RSA_0.5_3	0.448	0.505	0.484
	Swap	0.785	0.748	0.753

Tabla F.35: Resultados de *LSTM* con sus distintas versiones utilizando el *dataset ISSDA CER* Electricidad

Métrica	Escenario	LSTM_v1	LSTM_v2	LSTM_v3
tnr	Normal	0.493	0.566	0.513
tpr	FDI10	0.955	0.961	0.977
	FDI30	0.832	0.872	0.844
	Avg	0.534	0.385	0.576
	RSA_0.25_1.1	0.621	0.611	0.608
	RSA_0.5_3	0.245	0.342	0.303
	Swap	0.411	0.576	0.533

Tabla F.36: Resultados de *LSTM* con sus distintas versiones utilizando el *dataset ISSDA CER* Gas

Métrica	Escenario	LSTM_v1	LSTM_v2	LSTM_v3
tnr	Normal	0.873	0.874	0.859
tpr	FDI10	1.000	1.000	1.000
	FDI30	0.984	0.985	0.991
	Avg	0.976	0.890	0.974
	RSA_0.25_1.1	0.900	0.903	0.890
	RSA_0.5_3	0.636	0.670	0.603
	Swap	0.895	0.892	0.858

Tabla F.37: Resultados de *LSTM* con sus distintas versiones utilizando el *dataset Solar Ausgrid* para escenarios de consumo

Métrica	Escenario	LSTM_v1	LSTM_v2	LSTM_v3
tnr	Normal	0.996	0.995	0.992
tpr	Percentile	0.963	0.999	1.000
	Rating	0.917	0.978	0.897
	RSA_0.5_3	0.886	0.887	0.985

Tabla F.38: Resultados de *LSTM* con sus distintas versiones utilizando el *dataset Solar Ausgrid* para escenarios de generación

Pruebas tamaño de ventana

En la Sección 3.3.1 a partir de la Tabla 3.2, se concluyó que el tamaño óptimo de la ventana es de 5 días ya que obtiene los mejores resultados aunque sin muchas diferencias respecto a otros tamaños. A continuación, se muestran las métricas completas del modelo con diferentes ventanas para cada tipo de escenario y para cada conjunto de datos.

Métrica	**Escenario**	**LSTM_2**	**LSTM_3**	**LSTM_5**	**LSTM_7**	**LSTM_10**	**LSTM_14**	**LSTM_30**
tnr	Normal	0.779	0.771	0.770	0.762	0.766	0.764	0.768
tpr	FDI10	0.999	0.998	0.998	0.998	0.998	0.998	0.999
	FDI30	0.954	0.953	0.952	0.951	0.950	0.948	0.953
	Avg	0.889	0.885	0.882	0.882	0.881	0.879	0.878
	RSA_0.25_1.1	0.806	0.804	0.796	0.798	0.799	0.798	0.805
	RSA_0.5_3	0.503	0.503	0.514	0.490	0.514	0.495	0.454
	Swap	0.788	0.802	0.771	0.776	0.810	0.799	0.782

Tabla F.39: Resultados de *LSTM* con distintos tamaños de ventana utilizando el *dataset ISSDA CER* Electricidad

Métrica	**Escenario**	**LSTM_2**	**LSTM_3**	**LSTM_5**	**LSTM_7**	**LSTM_10**	**LSTM_14**	**LSTM_30**
tnr	Normal	0.511	0.542	0.545	0.561	0.577	0.599	0.574
tpr	FDI10	0.955	0.955	0.954	0.956	0.954	0.951	0.949
	FDI30	0.833	0.834	0.831	0.832	0.824	0.831	0.819
	Avg	0.620	0.619	0.619	0.562	0.463	0.509	0.524
	RSA_0.25_1.1	0.599	0.580	0.630	0.611	0.595	0.632	0.614
	RSA_0.5_3	0.263	0.280	0.352	0.271	0.330	0.290	0.355
	Swap	0.459	0.519	0.511	0.496	0.481	0.465	0.480

Tabla F.40: Resultados de *LSTM* con distintos tamaños de ventana utilizando el *dataset ISSDA CER* Gas

Métrica	**Escenario**	**LSTM_2**	**LSTM_3**	**LSTM_5**	**LSTM_7**	**LSTM_10**	**LSTM_14**	**LSTM_30**
tnr	Normal	0.860	0.858	0.858	0.852	0.851	0.853	0.860
tpr	FDI10	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	FDI30	0.977	0.977	0.977	0.977	0.977	0.977	0.977
	Avg	0.997	1.000	0.999	0.999	0.999	0.999	1.000
	RSA_0.25_1.1	0.882	0.878	0.881	0.878	0.878	0.880	0.886
	RSA_0.5_3	0.647	0.656	0.663	0.658	0.653	0.646	0.641
	Swap	0.826	0.830	0.828	0.827	0.837	0.834	0.831

Tabla F.41: Resultados de *LSTM* con distintos tamaños de ventana utilizando el *dataset Solar Ausgrid* para escenarios de consumo

Métrica	Escenario	LSTM_2	LSTM_3	LSTM_5	LSTM_7	LSTM_10	LSTM_14	LSTM_30
tnr	Normal	0.989	0.989	0.987	0.989	0.990	0.993	0.991
tpr	Percentile	0.998	0.998	0.999	0.999	0.999	0.999	0.997
	Rating	0.985	0.981	0.991	0.990	0.996	0.994	0.993
	RSA_0.5_3	0.883	0.883	0.874	0.870	0.871	0.875	0.879

Tabla F.42: Resultados de *LSTM* con distintos tamaños de ventana utilizando el *dataset Solar Ausgrid* para escenarios de generación

Optimización de hiperparámetros

Esta sección sirve como ampliación de la Sección 3.3.4 y la única diferencia de esta respecto a la sección del Anexo F.4.1 es que se muestra el fichero que guarda la información de la mejor combinación de los hiperparámetros para cada evaluación del modelo *LSTM*.

```
LSTM (meter_id = 8, window = 2, dataset = solar_generation)
{'n_layers': 2, 'batch_size': 56, 'LSTM_units': 288, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0014'}
================================================
LSTM (meter_id = 50, window = 2, dataset = solar_consumption)
{'n_layers': 1, 'batch_size': 8, 'LSTM_units': 384, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0012'}
================================================
LSTM (meter_id = 1156, window = 2, dataset = gas)
{'n_layers': 0, 'batch_size': 8, 'LSTM_units': 384, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0015'}
================================================
LSTM (meter_id = 3302, window = 2, dataset = electricity)
{'n_layers': 3, 'batch_size': 200, 'LSTM_units': 416, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 1, 'tuner/round': 1, 'tuner/trial_id': '0018'}
================================================
LSTM (meter_id = 160, window = 2, dataset = solar_consumption)
{'n_layers': 3, 'batch_size': 8, 'LSTM_units': 96, 'tuner/epochs': 15, 'tuner/initial_epoch': 0, 'tuner/bracket': 0, 'tuner/round': 0}
================================================
LSTM (meter_id = 1648, window = 2, dataset = electricity)
{'n_layers': 0, 'batch_size': 56, 'LSTM_units': 128, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 1, 'tuner/round': 1, 'tuner/trial_id': '0018'}
================================================
LSTM (meter_id = 241, window = 2, dataset = solar_generation)
{'n_layers': 2, 'batch_size': 24, 'LSTM_units': 128, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0012'}
================================================
LSTM (meter_id = 1244, window = 2, dataset = gas)
{'n_layers': 1, 'batch_size': 8, 'LSTM_units': 224, 'tuner/epochs': 15, 'tuner/initial_epoch': 0, 'tuner/bracket': 0, 'tuner/round': 0}
================================================
LSTM (meter_id = 229, window = 2, dataset = solar_consumption)
{'n_layers': 0, 'batch_size': 88, 'LSTM_units': 288, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 1, 'tuner/round': 1, 'tuner/trial_id': '0018'}
================================================
LSTM (meter_id = 208, window = 2, dataset = solar_generation)
{'n_layers': 0, 'batch_size': 8, 'LSTM_units': 416, 'tuner/epochs': 15, 'tuner/initial_epoch': 5, 'tuner/bracket': 2, 'tuner/round': 2, 'tuner/trial_id': '0014'}
================================================
LSTM (meter_id = 1996, window = 2, dataset = gas)
{'n_layers': 3, 'batch_size': 24, 'LSTM_units': 512, 'tuner/epochs': 15, 'tuner/initial_epoch': 0, 'tuner/bracket': 0, 'tuner/round': 0}
================================================
```

Figura F.2: Fichero que recoge los resultados del proceso de optimización de hiperparámetros de *LSTM*