

Universidad de Alcalá

Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

**Detección y predicción de errores
en smart grids mediante modelos
de inteligencia artificial**

**ESCUELA POLITÉCNICA
SUPERIOR**

Autor: Paula Bartolomé Mora

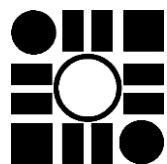
Tutor: Elisa Rojas Sánchez

Cotutor: David Carrascal Acebron

2024



Máster Universitario en Ingeniería de Telecomunicación



Universidad
de Alcalá

Madrid, 1 de julio de 2024

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

**Detección y predicción de errores en smart grids mediante
modelos de inteligencia artificial**

Autor: Paula Bartolomé Mora

Tutor: Elisa Rojas Sánchez

Cotutor: David Carrascal Acebron

Tribunal:

Presidente: Isaías Martínez Yelmo

Vocal 1º: Carlos Santos Pérez

Vocal 2º: Juan Antonio Carral Pelayo

A mi familia siempre.

Agradecimientos

Me gustaría comenzar agradeciendo y reconociendo el trabajo de mis tutores, Elisa Rojas y David Carrascal. Gracias, Elisa, por acompañarme desde que inicié mi Trabajo de Fin de Grado y por darme la oportunidad de volver a desarrollar este proyecto contigo. Gracias, David, por tu apoyo incondicional, por tus consejos, por tantos buenos momentos que hemos compartido y por acompañarme en este camino, que no ha sido fácil. Sin duda, este proyecto no habría sido lo mismo sin ti.

Con este proyecto cierro mi etapa universitaria y no puedo dejar de agradecer a todas las personas que me llevó de la misma. Todo lo que me han aportado y todas las experiencias que he podido compartir con ellos no tiene precio. Gracias de verdad por darme un recuerdo tan bonito de la universidad.

Y, por supuesto, tengo que terminar agradeciendo a mi familia por ser mi pilar principal y por creer en mí en cada paso de mi vida. Gracias.

Resumen

En este Trabajo Fin de Máster (TFM) se presenta el diseño y desarrollo de diferentes modelos de *Machine Learning* (ML) y *Deep Learning* (DL) para la detección y predicción de errores en entornos de *Smart Grids* (SG).

En este proceso se utilizan datos de una implementación de *Smart Grid* (SG) real, los cuales requieren un procesamiento exhaustivo. Posteriormente, se generan una serie de escenarios y topologías aleatorias con la herramienta BRITE y se aplican modificaciones en el funcionamiento del algoritmo DEN2NE. Se pretende detectar y predecir errores sobre las rutas que se obtienen en el mismo y, por ello, se ejecutan múltiples simulaciones que permitan identificar los patrones de error sobre los que entrenar los modelos de forma precisa. El proyecto concluye con el análisis y evaluación de los resultados obtenidos de los modelos para definir el que proporciona un rendimiento óptimo.

Palabras clave: [Smart Grids](#); [Machine Learning](#); [Deep Learning](#); [Big Data](#); [Prosumidor](#); [IoT](#);

Abstract

In this Master's Thesis (TFM), we present the design and development of various Machine Learning (ML) and Deep Learning (DL) models for detecting and predicting errors in Smart Grid (SG) environments.

This process involves using real implementation data, which requires comprehensive pre-processing. Subsequently, a series of scenarios and random topologies are generated using the BRITE tool, and modifications are applied to the DEN2NE algorithm's operation. The aim is to detect and predict errors in the routes obtained, and therefore, multiple simulations are run to identify error patterns to accurately train the models. The project concludes with the analysis and evaluation of the results obtained from the models to determine the one that offers optimal performance.

Keywords: Smart Grids; Machine Learning; Deep Learning; Big Data; Prosumer; IoT;

“Cualquier tecnología lo suficientemente avanzada es indistinguible de la magia”

Arthur C. Clarke.

Índice general

Resumen	v
Abstract	vii
1. Introducción	1
1.1. Presentación	1
1.2. Objetivos	2
1.3. Estructura del TFM	3
2. Estado del arte	7
2.1. Smart Grids	7
2.1.1. Flujo energético y gestión bidireccional	9
2.1.2. Prosumidores y microgrids	10
2.1.2.1. Mercado energético	11
2.1.3. Estructura de una Smart Grid	14
2.1.3.1. Generación	14
2.1.3.2. Distribución	17
2.1.3.3. Consumo	21
2.1.4. Seguridad en Smart grids	22
2.1.5. Desafíos futuros de las smart grids	23
2.2. IoT	24
2.2.1. IoE	25
2.3. DEN2NE	26
2.4. Big Data	29
2.4.1. Computación en nube	31
2.5. Inteligencia Artificial	33
2.5.1. Machine Learning (ML)	33
2.5.1.1. Random Forest (RF)	35
2.5.1.2. Support Vector Machines (SVM)	37
2.5.2. Deep Learning (DL)	41
2.5.2.1. Red Neuronal Artificial (ANN) y Perceptrón Multicapa (MLP)	42

2.6. Herramientas software	45
2.6.1. BRITE	45
2.6.1.1. Definición de topologías	45
2.6.1.2. Modelos de topologías	46
2.6.1.3. Automatización de la ejecución	48
3. Diseño y análisis de datos	51
3.1. Análisis de los <i>datasets</i> disponibles	51
3.1.1. <i>SustDataED</i>	55
3.1.2. Estructura del dataset <i>SustDataED</i>	55
3.1.2.1. Medidas de consumo de energía	57
3.1.2.2. Medidas de eventos de potencia	58
3.1.2.3. Medidas demográficas	59
3.1.2.4. Medidas de producción de energía	59
3.1.2.5. Medidas de eventos de usuario	60
3.1.2.6. Medidas de condiciones ambientales y climáticas	61
3.1.3. Conclusiones del análisis del dataset <i>SustDataED</i>	61
3.2. Simulación de datos de producción	62
3.2.1. Estudio y análisis de la ubicación (<i>Global Solar Atlas</i>)	63
3.2.1.1. Identificación de capas de datos	63
3.2.1.2. Análisis de resultados de radiación	66
3.2.1.3. Configuración y cuantificación del potencial energético de la ubicación	68
3.2.1.4. Análisis de resultados de potencia	68
3.2.2. Creación del dataset de producción (<i>PVWatts</i>)	71
3.2.2.1. Configuración de los parámetros de entrada	71
3.2.2.2. Configuración de las pérdidas del sistema real	73
3.2.2.3. Análisis de resultados de la simulación	75
3.3. Procesado de los datos	79
3.3.1. Procesado de datos de consumo	80
3.3.1.1. Análisis de la situación inicial y primeros pasos	80
3.3.1.2. Síntesis de la información	83
3.3.1.3. Automatización del proceso de síntesis de la información .	84
3.3.1.4. Selección del despliegue	89
3.3.1.5. Selección del rango temporal y procesamiento adicional .	89
3.3.2. Procesado de datos de producción	92
3.3.2.1. Preprocesamiento de datos de <i>SustDataED</i>	92
3.3.2.2. Preprocesamiento de datos de <i>PVWatts</i>	96

3.3.2.3. Selección de datos de producción	100
3.3.3. Combinación de datos de consumo y de producción	101
3.4. Planteamiento de escenarios y generación de topologías en BRITE	105
3.4.1. Planteamiento de escenarios y configuración de BRITE	105
3.4.2. Resultados de la generación de topologías aleatorias	106
3.5. Simulación de las topologías en DEN2NE	107
3.5.1. Adaptación del algoritmo DEN2NE a las pruebas	108
3.5.1.1. Importación de los perfiles de carga reales	108
3.5.1.2. Introducción de la etiqueta de error	109
3.5.1.3. Extracción de resultados	110
3.5.1.4. Configuración de la capacidad de los enlaces	111
3.5.2. Configuración de los parámetros de entrada	111
3.5.3. Creación del dataset final y conclusiones de las pruebas	113
4. Desarrollo y evaluación del modelo	117
4.1. Preparación al desarrollo	117
4.2. Técnicas de ML	119
4.2.1. Random Forest (RF)	120
4.2.1.1. Puntuación de características	120
4.2.1.2. Optimización de hiperparámetros	122
4.2.1.3. Selección de características	124
4.2.1.4. Ejecución del modelo y evaluación de resultados	127
4.2.2. Support Vector Machines (SVM)	132
4.2.2.1. Puntuación de características	132
4.2.2.2. Optimización de hiperparámetros	134
4.2.2.3. Selección de características	136
4.2.2.4. Ejecución del modelo y evaluación de resultados	136
4.3. Técnicas de DL	138
4.3.1. Redes Neuronales Artificiales (ANN)	138
4.3.1.1. Optimización de hiperparámetros y ejecución del modelo de ANN de <i>sklearn</i>	139
4.3.1.2. Ejecución del modelo de ANN de <i>keras</i> y evaluación de resultados	143
5. Conclusiones y trabajo futuro	147
5.1. Conclusiones del TFM	147
5.2. Líneas de trabajo futuro	150

Bibliografía	151
Lista de Acrónimos y Abreviaturas	161
A. Anexo I - Pliego de condiciones	165
A.1. Condiciones materiales y equipos	165
A.1.1. Especificaciones Máquina A	165
A.1.2. Especificaciones Máquina B	165
A.1.3. Especificaciones Máquina C	166
A.2. Virtualización del sistema operativo Linux	167
B. Anexo II - Presupuesto	169
B.1. Duración del proyecto	169
B.2. Costes del proyecto	169
C. Anexo III - Manuales de usuario e Instalación	173
C.1. Herramienta BRITE	173
C.1.1. Compilación e instalación de BRITE	173
C.1.2. Ejecución de BRITE	174

Índice de figuras

1.1.	Secuencia de acciones para la detección y predicción de errores	5
2.1.	Red de distribución y flujo bidireccional de datos y energía en una <i>smart grid</i> [10]	8
2.2.	Infraestructura de una interfaz AMI para el flujo bidireccional de energía y datos [8]	10
2.3.	Estructura de la respuesta a la demanda en el contexto de una SG con prosumidores [13]	11
2.4.	Modelo basado en las interacciones de los agregadores con los prosumidores y el mercado eléctrico [15]	12
2.5.	Estrategia de DSM basada en interacciones individuales de cada consumidor con la compañía [19]	13
2.6.	Estrategia de DSM para SGs basada en interacciones entre los usuarios y la compañía [19]	14
2.7.	Estructura de componentes de una SG [6]	15
2.8.	Esquema de funcionamiento de una unidad CHP [20]	16
2.9.	Gestión de operaciones en una unidad CHP [21]	16
2.10.	Representación del proceso P2G [22]	17
2.11.	Instalación de un SVC en Sylling (Noruega) [25]	18
2.12.	Representación de un sistema de transmisión de energía eólica con líneas HVDC [29]	21
2.13.	Infraestructura de componentes de un sistema IoT [39]	25
2.14.	Ejemplo del proceso de balance de recursos [3]	27
2.15.	Proceso de asignación de etiquetas [3]	28
2.16.	Proceso de distribución energética [3]	29
2.17.	Infraestructura de <i>Big Data</i> enfocada al ámbito de las SGs [30]	31
2.18.	Plataforma de análisis de <i>Big Data</i> en el entorno de SGs [43]	32
2.19.	Categorías de ML [50]	34
2.20.	Modelo RF [51]	36
2.21.	Cuantificación del hiperplano óptimo entre dos clases en el SVM [54]	38

2.22. Representación de vectores clasificados erróneamente dentro de los márgenes establecidos en el SVM [56]	39
2.23. Configuración del parámetro de regularización C [58]	40
2.24. Aplicación del kernel RBF [59]	41
2.25. Diferencias estructurales entre el ML y el DL [60]	42
2.26. Arquitectura de un MLP [62]	43
2.27. Funciones de activación [64]	44
2.28. Modelos soportados por BRITE [4]	46
2.29. Ejemplo de topología Router Waxman	47
2.30. Ejemplo de topología Router Barabasi-Albert	48
2.31. Esquema de funcionamiento de BRITE y del <i>parser</i> [3]	49
 3.1. Representación del período temporal que comprende el proceso de recolección de datos para cada hogar	56
3.2. Gráfica de valores de múltiples fuentes energéticas (<i>SustDataED</i>)	60
3.3. Mapa mundial del índice de radiación directa normal (DNI) mundial [83] . .	65
3.4. Mapa del índice de radiación horizontal global (GHI) mundial [83]	65
3.5. Mapa del potencial de producción energética fotovoltaica (PVOUT) mundial [83]	66
3.6. Mapa del índice de radiación directa normal (DNI) de Madeira [83]	67
3.7. Mapa del índice de radiación horizontal global (GHI) de Madeira [83] . .	67
3.8. Mapa del potencial de producción energética fotovoltaica (PVOUT) de Madeira [83]	67
3.9. Representación de la trayectoria solar percibida en la localización de la ciudad de Funchal [83]	69
3.10. Comparación de valores totales mensuales de producción energética fotovoltaica (PVOUT) respecto a la radiación directa normal (DNI) [83]	69
3.11. Perfiles promedios de potencial de producción energética fotovoltaica (PVOUT) y de radiación directa normal (DNI) [83]	70
3.12. Representación de las diferentes opciones de array que permite configurar la herramienta <i>PVWatts</i> [88]	73
3.13. Gráfica para el cálculo de las pérdidas por sombreado en la herramienta <i>PVWatts</i> [88]	74
3.14. Comparación de valores totales mensuales de producción energética fotovoltaica (PVOUT) respecto a la radiación directa normal (DNI) a partir de los resultados de la simulación	76
3.15. Salida de <i>datasamples_onenode.ipynb</i> tras aplicar la operación de promedio	84
3.16. Lanzamiento de pruebas con <i>threads</i>	87

3.17. Lanzamiento de pruebas con procesos	87
3.18. Representación del período temporal que comprende el proceso de recolección de datos para los hogares del segundo despliegue	90
3.19. Comparación del consumo de las viviendas 5 y 17 en el rango seleccionado (<i>SustDataED</i>)	91
3.20. Comparación del consumo de las viviendas 13 y 16 en el rango seleccionado (<i>SustDataED</i>)	92
3.21. Gráfica de energía solar producida a nivel global (<i>SustDataED</i>)	93
3.22. Gráfica de temperatura en el rango seleccionado (<i>SustDataED</i>)	94
3.23. Gráfica de energía solar producida en el rango seleccionado (<i>SustDataED</i>) .	94
3.24. Correlación entre temperatura y energía solar producida en el mes de agosto (<i>SustDataED</i>)	95
3.25. Representación de los valores de correlación mediante un mapa de calor (<i>SustDataED</i>)	96
3.26. Gráfica de energía solar producida en el rango seleccionado (<i>PVWatts</i>) . .	97
3.27. Representación de los valores de correlación mediante un mapa de calor (<i>PVWatts</i>)	98
3.28. Correlación entre temperatura y energía solar producida en el mes de agosto (<i>PVWatts</i>)	99
3.29. Correlación entre radiación y energía solar producida en el mes de agosto (<i>PVWatts</i>)	99
3.30. Representación de los valores de correlación mediante gráficas (<i>PVWatts</i>) .	100
3.31. Comparación entre el consumo y la generación de potencia en la vivienda 1 en el rango seleccionado	102
3.32. Gráfica de la carga neta resultante en la vivienda 1 en el rango seleccionado	102
3.33. Diagrama completo de diseño y procesamiento para la obtención de los ficheros <i>load_x.csv</i>	104
3.34. Representación de diferencias de los modelos Router Barabasi-Albert y Waxman a partir de los mismos parámetros de entrada	107
3.35. Diagrama completo de diseño del dataset final	113
3.36. Nomenclatura de los directorios y ficheros de resultados	114
4.1. Puntuación de características del RF mediante el atributo <i>feature_importances_</i>	121
4.2. Puntuación de características del RF mediante el método <i>permutation_importance</i>	122
4.3. Análisis de la precisión del modelo en función del número de características empleadas	125
4.4. Análisis de la varianza en función del número de componentes empleadas en el PCA	127

4.6. Matriz de confusión [101]	128
4.5. Ejemplo gráfico de un estimador o árbol del RF [100]	129
4.7. Puntuación de características del SVM mediante los atributos <i>support_vectors_y dual_coef_</i>	133
4.8. Puntuación de características del SVM mediante el método <i>permutation_importance</i>	134
4.9. Representación del valor de precisión en función de las iteraciones para el modelo de MLP escogido	142
4.10. Representación del valor de la función de pérdidas en función de las iteraciones para el modelo de MLP escogido	142
4.11. Representación de la estructura de capas de la ANN [104]	143
4.12. Representación del valor de precisión en función de las iteraciones para el modelo de ANN	145
4.13. Representación del valor de la función de pérdidas en función de las iteraciones para el modelo de ANN	145
A.1. Especificaciones de la máquina A	166
A.2. Especificaciones de la máquina B	166
A.3. Características del sistema operativo	167
B.1. Diagrama de Gantt del proyecto	171

Índice de tablas

2.1.	Tecnologías FACTS de 1 ^a y 2 ^a generación	19
3.1.	Comparación entre datasets públicos en el ámbito NILM [69] [80] [81] [70] . .	54
3.2.	Tabla de medidas de consumo energético [78]	57
3.3.	Tabla de medidas de eventos de potencia [78]	59
3.4.	Tabla de medidas de producción de energía [78]	59
3.5.	Tabla de medidas de producción de energía [78]	60
3.6.	Tabla de medidas de eventos de usuario [78]	61
3.7.	Tabla de medidas de condiciones ambientales [78]	61
3.8.	Tabla de valores extraídos para cada índice de radiación en la ciudad de Funchal [83]	66
3.9.	Dataset con valores totales mensuales [88]	75
3.10.	Dataset con muestras adquiridas por hora [88]	75
3.11.	Tabla de comparación de los valores totales mensuales de producción energética fotovoltaica (PVOUT) y de radiación directa normal (DNI) de las dos herramientas (valores teóricos y simulados)	78
3.12.	Resumen de los datos de consumo del dataset <i>SustDataED</i> [78]	80
3.13.	Resumen de las características de los ficheros de los despliegues de <i>SustDataED</i> .	81
3.14.	Estructura de campos de cada fichero <i>consum_x.csv</i>	88
3.15.	Estructura de campos del fichero <i>mean_prod.csv</i>	93
3.16.	Dataset resultante de la etapa de procesamiento	103
3.17.	Configuraciones de enlaces de <i>links_config.csv</i>	111
3.18.	Dataset final obtenido por cada instante temporal probado en DEN2NE . .	116
4.1.	Resultados extraídos del atributo <i>cv_results_</i> del <i>Grid Search</i> en el RF . .	124
4.2.	Resultados de aplicación de la matriz de confusión al RF	131
4.3.	Resultados de aplicación del <i>K-Fold Cross Validation</i> al <i>Random Forest</i> (RF)	132
4.4.	Resultados extraídos del atributo <i>cv_results_</i> del SVM	135
4.5.	Resultados de aplicación de la matriz de confusión al SVM	137
4.6.	Resultados de aplicación del <i>K-Fold Cross Validation</i> al SVM	138

4.7. Resultados de precisión (%) (<i>mean_test_score</i>) extraídos del atributo <i>cv_results_</i> del MLP	140
4.8. Resultados de desviación típica (%) (<i>std_test_score</i>) extraídos del atributo <i>cv_results_</i> del MLP	141
4.9. Resultados de tiempo (s) (<i>mean_fit_time</i>) extraídos del atributo <i>cv_results_</i> del MLP	141
4.10. Resultados de aplicación de la matriz de confusión a las ANNs	146
4.11. Resultados de aplicación de los métodos <i>score</i> y <i>evaluate</i> a las <i>Artificial Neural Network</i> (ANN)s	146
5.1. Síntesis de resultados obtenidos del desarrollo de los modelos de ML y DL .	149
B.1. Promedio de horas de trabajo	169
B.2. Presupuesto desglosado del hardware para el TFM	170
B.3. Presupuesto desglosado del software para el TFM	170
B.4. Presupuesto total con IVA	170

Índice de Códigos

3.1.	Formato del timestamp	81
3.2.	Eliminación de valores NaN	82
3.3.	Fragmentación de ficheros iniciales	82
3.4.	Función de lectura de los ficheros	84
3.5.	Función de cálculo del promedio	85
3.6.	Función de automatización del cálculo del promedio	86
3.7.	Creación de procesos	88
3.8.	Aplicación del rango temporal a los ficheros	90
3.9.	Combinación de datos de clima y de producción	93
3.10.	Configuración de los parámetros de entrada en el script de automatización de BRITE	106
3.11.	Configuración de los parámetros de entrada en el script de automatización de DEN2NE	112
3.12.	Combinación de ficheros de resultados y de test	115
4.1.	Codificación de los datos categóricos	118
4.2.	Codificación de los datos categóricos	118
4.3.	Estandarización de los subconjuntos	119
4.4.	Clasificador RF por defecto	120
4.5.	Cuadrícula de parámetros RF	122
4.6.	Construcción del objeto <i>GridSearchCV()</i>	123
4.7.	Aplicación del RFECV	124
4.8.	Resultados del RFECV	124
4.9.	Aplicación del <i>kBest</i>	126
4.10.	Resultados del <i>kBest</i> para $k=8$	126
4.11.	Clasificador RF seleccionado	127
4.12.	Implementación de la matriz de confusión	130
4.13.	Implementación del <i>K-Fold Cross Validation</i>	131
4.14.	Clasificador SVM por defecto	132
4.15.	Cuadrícula de parámetros SVM	134
4.16.	Cuadrícula de parámetros MLP	139

4.17. Clasificador MLP por defecto	139
4.18. Clasificador MLP óptimo	141
4.19. Definición del modelo de ANN de Keras	143
4.20. Entrenamiento del modelo de ANN de Keras	144
C.1. Instalación de paquetes	173
C.2. Ejecución de la interfaz gráfica de BRITE	174
C.3. Ejecucion de BRITE por la línea de comandos	174

1. Introducción

En este primer capítulo, se presentarán a modo de introducción los aspectos más relevantes del Trabajo Fin de Máster (TFM) desarrollado, como son la transformación de las redes energéticas en nuevas redes inteligentes de energía (del inglés *Smart Grids* (SG)) y su consecuente necesidad de monitorización, además del creciente auge de la Inteligencia Artificial (IA) y de las técnicas de aprendizaje automático y profundo (del inglés *Machine Learning* (ML) y *Deep Learning* (DL)).

De la misma manera, se hará hincapié en los objetivos que se establecen para este TFM, describiendo así la secuencia de acciones que se seguirá para poder llevarlos a cabo. Dichos objetivos serán los pilares sobre los que se sustente el diseño y el desarrollo de modelos eficientes basado en ML y DL para la detección y predicción de errores en SGs. De forma adicional, se incluirá un último apartado en referencia a la estructura de capítulos que seguirá este TFM, así como una explicación breve sobre los temas que se abordarán en cada uno de ellos.

1.1. Presentación

En los últimos años, el concepto de red eléctrica convencional está siendo completamente transformado como resultado del creciente desarrollo de las SGs [1] [2]. La posibilidad de brindar a tiempo real una monitorización del proceso de distribución eléctrica y a su vez, integrar una participación activa del consumidor, convierte a estas redes en uno de los pilares de la transición hacia la descarbonización.

Por lo tanto, se ha convertido en un objetivo primordial el diseño de nuevos protocolos que provean un encaminamiento y una distribución energética eficientes. En este contexto, desde el equipo de investigación NetIS de la *Universidad de Alcalá* (UAH) se ha desarrollado el algoritmo *Distributed ENergy ENvironments and Networks* (DEN2NE) [3], con el fin de establecer un reparto colaborativo de los recursos en redes densas.

Teniendo esto en cuenta, se puede expresar que DEN2NE establece la posibilidad de aplicación al caso de uso de las SGs y permite realizar una distribución óptima de los recursos energéticos de los nodos que forman parte de una topología de red para conseguir finalmente un balance a nivel global. Sin embargo, el algoritmo se encuentra actualmente con un desafío a afrontar, basado en la imperiosa necesidad de poder detectar y predecir de una forma precisa los posibles errores que se puedan producir en el proceso de distribución energética entre nodos. En caso de fallos en la red, DEN2NE debería tener la capacidad de responder de una forma rápida y específica, pues es de vital importancia tomar acciones a nivel de distribución para evitar que los errores se agraven y escalen al resto de la red.

1.2. Objetivos

Tomando en consideración el apartado anterior, se define como objetivo principal de este TFM el desarrollo de diversas técnicas de ML y DL para identificar de forma precisa los posibles errores que se pueden producir en una SG durante el proceso de distribución energética que, como se ha introducido, se aplicará mediante el algoritmo DEN2NE.

En otros términos, la identificación de patrones se apoyará en el análisis de grandes volúmenes de datos provenientes de implementaciones de SGs reales. Estos serán previamente procesados de una forma exhaustiva para reducir su tamaño y seleccionar la información verdaderamente necesaria para el posterior desarrollo de los modelos, a partir de los que se podrá determinar un modelo final de predicción de fallos para entornos de SGs. Es preciso indicar que los ficheros desarrollados en el contexto de este TFM estarán disponibles en el repositorio¹ de GitHub dedicado a este proyecto.

A modo gráfico, en la Figura 1.1 se representa el diagrama de flujo que define la secuencia de acciones a realizar. Entrando en detalle, se pueden definir los diferentes objetivos en los siguientes puntos:

- **Estudio del estado del arte y análisis de las diferentes fuentes de datos reales disponibles.** Se realizará un estudio en profundidad del contexto en el que se engloban las SGs y los protocolos de distribución de recursos, poniendo el enfoque en DEN2NE. Será necesaria una investigación sobre la disponibilidad de fuentes de datos reales y una evaluación de su viabilidad para este TFM. De forma adicional, se precisará del estudio y análisis de herramientas de simulación de datos.

¹<https://github.com/PaulaBartolomeMora/TFM>

- **Definición y procesado del volumen de datos para el posterior desarrollo.** El procesamiento de los datos adquiridos será fundamental para simplificar su comprensión. Además, se llevará a cabo la selección de la cantidad de información realmente útil para el desarrollo.
- **Planteamiento de escenarios y generación de diferentes topologías.** A partir del volumen final de datos que se elaborará, se plantearán una serie de escenarios mediante la generación de topologías aleatorias de nodos con la herramienta *Boston University Representative Internet Topology Generator* (BRITE) [4].
- **Simulación de topologías y extracción del conjunto de datos final.** Las topologías generadas se importarán a un simulador propietario realizado en python [5]. Por consiguiente, se entrará en el funcionamiento del algoritmo DEN2NE y se realizarán las modificaciones necesarias para que, mediante la ejecución del mismo, se pueda extraer el conjunto de datos final sobre el que se va a desarrollar.
- **Desarrollo y entrenamiento de diferentes modelos de ML y DL.** Se desarrollarán múltiples modelos de predicción de errores a partir del empleo de diversas técnicas de ML y DL. Por consiguiente, se llevará a cabo el entrenamiento de los mismos, tomando como base el conjunto de datos diseñado anteriormente e implementando una serie de técnicas de selección de los parámetros más significativos.
- **Análisis y evaluación de los resultados obtenidos.** Se analizarán los resultados de detección y predicción de errores de los diferentes modelos desarrollados. Mediante el análisis en cuestión, se determinará de forma concluyente cuál aporta una mayor efectividad para poder diagnosticar con un alto porcentaje de precisión los errores que se producen en una SG.

1.3. Estructura del TFM

En esta sección se expone de forma general la estructura de capítulos que tendrá la memoria del presente TFM con una breve descripción del contenido que abarcarán cada uno de ellos.

Capítulo 1: Introducción. Se comenzará esta memoria con una presentación de los aspectos más significativos. Se expondrán las motivaciones que han originado la realización de este TFM y los objetivos que se pretenden alcanzar con el mismo.

Capítulo 2: Estado del arte. Se establecerá el marco teórico en el que se sitúa este TFM y se documentarán los conceptos principales para introducir en un contexto

lo suficientemente consistente el diseño y desarrollo práctico posterior. Además, se expondrá de forma detallada el funcionamiento del algoritmo DEN2NE.

Capítulo 3: Diseño y análisis de datos. Se llevará a cabo un análisis sobre las posibilidades de diseño, realizando una investigación exhaustiva sobre las fuentes de datos de implementaciones reales existentes y estudiando y ejecutando algunas herramientas de simulación para extraer información adicional. Después, se definirá todo el procesamiento que deberá llevarse a cabo sobre los datos para sintetizar los mismos y seleccionar únicamente la información útil. Las siguientes etapas vendrán definidas por el planteamiento de diferentes escenarios de red y la generación de topologías mediante la herramienta BRITE y el algoritmo DEN2NE, respectivamente. Finalmente, se obtendrá el conjunto de datos final sobre el que se entrenarán los modelos de ML y DL a desarrollar

Capítulo 4: Desarrollo y evaluación de los modelos. Se describirá en detalle el desarrollo de las técnicas de ML y DL que han sido seleccionadas según el contexto en el que se engloba este TFM. Los modelos obtenidos serán entrenados, analizados y, por consiguiente, evaluados. Los resultados definirán finalmente el modelo más apropiado para diagnosticar con alta precisión los errores que se pueden producir en el entorno de una SG.

Capítulo 5: Conclusiones y trabajo futuro. Se finalizará la memoria con un capítulo enfocado a las conclusiones obtenidas tras la realización de este TFM. Además, se indicarán las posibles vías de trabajo a futuro.

Bibliografía. Se incluirán todas las referencias de artículos, libros, páginas web u otros materiales que han sido consultados y empleados para elaborar esta memoria. Se seguirá el estilo de citación del *Institute of Electrical and Electronics Engineers* (IEEE), así como las recomendaciones oficiales de la normativa sobre TFMs de la UAH.

Anexos. Se añadirán los manuales de usuario y de instalación de herramientas. También, se expondrán las especificaciones del *hardware* sobre el que se ha desarrollado este TFM y una estimación de su presupuesto.

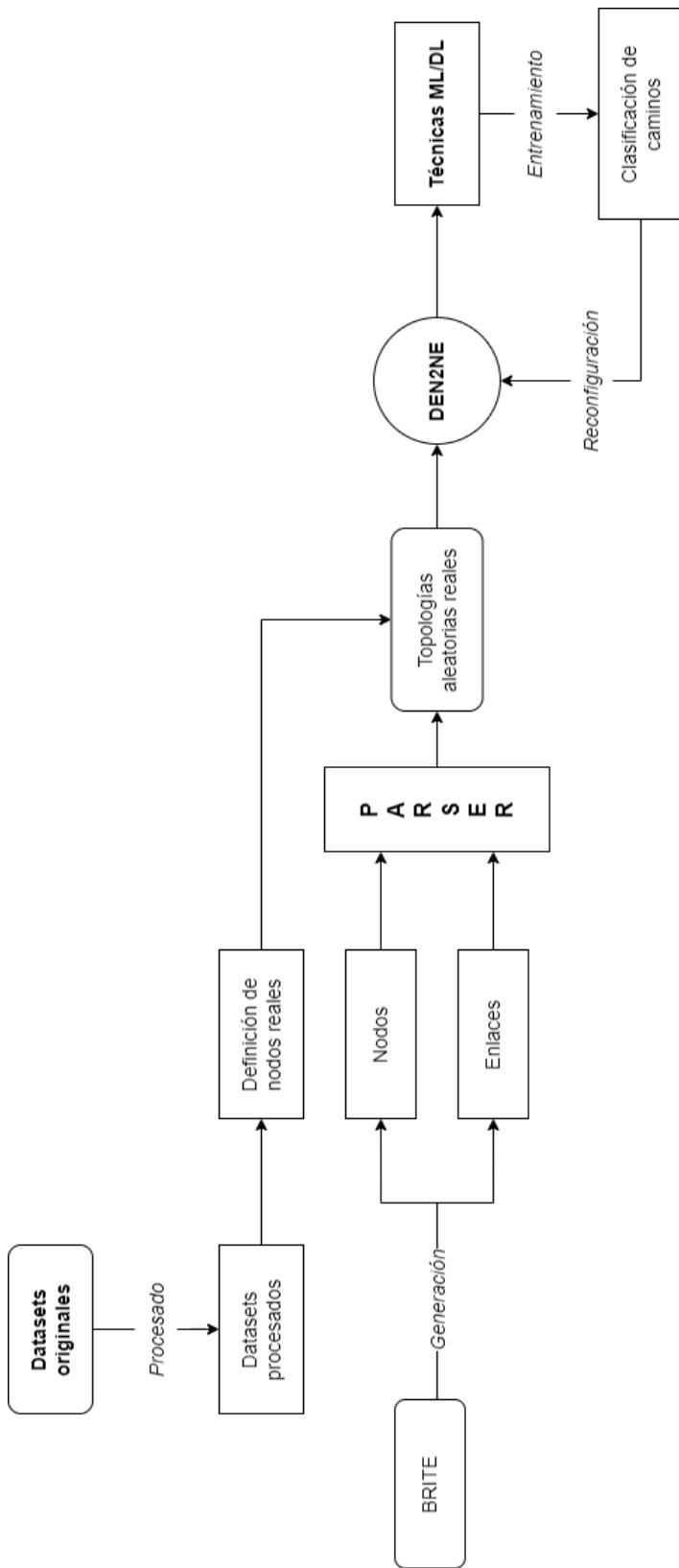


Figura 1.1: Secuencia de acciones para la detección y predicción de errores

2. Estado del arte

En el presente capítulo se entrará en profundidad en los conceptos más relevantes relacionados con el TFM. Por ello, el estado del arte se dividirá en una serie de Secciones enfocadas, por un lado, en el contexto de las SGs y las redes IoT (ver Secciones 2.1 y 2.2) y, por otro lado, en la descripción teórica del concepto de *big data* y de las técnicas de ML y DL (ver Secciones 2.4 y 2.5) que se aplicarán en la etapa de desarrollo.

De la misma forma, para aportar una buena compresión de los pasos de diseño que serán necesarios para llevar a cabo el planteamiento de escenarios de red, la generación de topologías y la ejecución de simulaciones, se añadirán las Secciones 2.3 y 2.6. La primera, expondrá las características que presenta el algoritmo DEN2NE y la segunda, el funcionamiento de la plataforma BRITE. Es por ello que se puede expresar que el propósito principal de este Capítulo se centra en establecer un marco teórico consistente y que, por tanto, permita conseguir una suficiente compresión del contexto, previamente a la exposición de las etapas de diseño y desarrollo del TFM, en los Capítulos 3 y 4.

2.1. Smart Grids

En las últimas décadas, se ha observado una transformación integral del modelo asociado a la red eléctrica convencional. El aumento de la energía demandada por los usuarios finales y los requisitos cada vez mayores de la industria ha tenido como consecuencia que algunos países hayan intentado diseñar redes eléctricas agrupadas en un conjunto de grandes redes nacionales. De este modo, todas las fuentes energéticas disponibles pueden estar conectadas para ser gestionadas conjuntamente en función de la demanda recibida, además de conseguir una coordinación a alto nivel [6].

Sin embargo, una red eléctrica no se puede definir como una entidad única e independiente, pues se trata de una agregación de redes, compañías energéticas y operadores que trabajan en distintos niveles de comunicación. Es por ello que la idea de desarrollar una gran red nacional encuentra cierto equilibrio energético, pero no llega a los altos porcentajes de eficiencia que pueden proveer las redes inteligentes energéticas o de otra forma, las *Smart Grids* (SG).

La iniciativa sobre redes inteligentes del IEEE, califica a las *smart grids* [7] como “imperativas y revolucionarias que implican nuevas capacidades de comunicación y control, fuentes de energía, modelos de generación y adherencia a estructuras regulatorias transjurisdicciones”. Por otro lado, en el año 2009, el Departamento de Energía de Estados Unidos [8] determinó en su reporte anual sobre SGs los siguientes objetivos principales que se perseguían con el desarrollo de este tipo de redes:

- Permitir una participación activa de los clientes en el sistema.
- Integrar todas las opciones de generación y almacenamiento de energía.
- Ofertar nuevos productos y servicios.
- Proporcionar energía de calidad y satisfacer un gran rango de necesidades.
- Optimizar el uso de los recursos y aumentar la eficiencia.
- Dar una respuesta rápida ante emergencias y problemas producidos en la red.

Tomando en consideración estos objetivos, se puede definir una SG [9] como una red inteligente con la capacidad de distribuir los suministros de energía de forma optimizada a los usuarios, basándose en la información que recoge de los mismos. Es por ello que supone una actualización digital de las redes de distribución y transmisión a larga distancia para incorporar sistemas de monitorización y control a tiempo real (ver Figura 2.2).

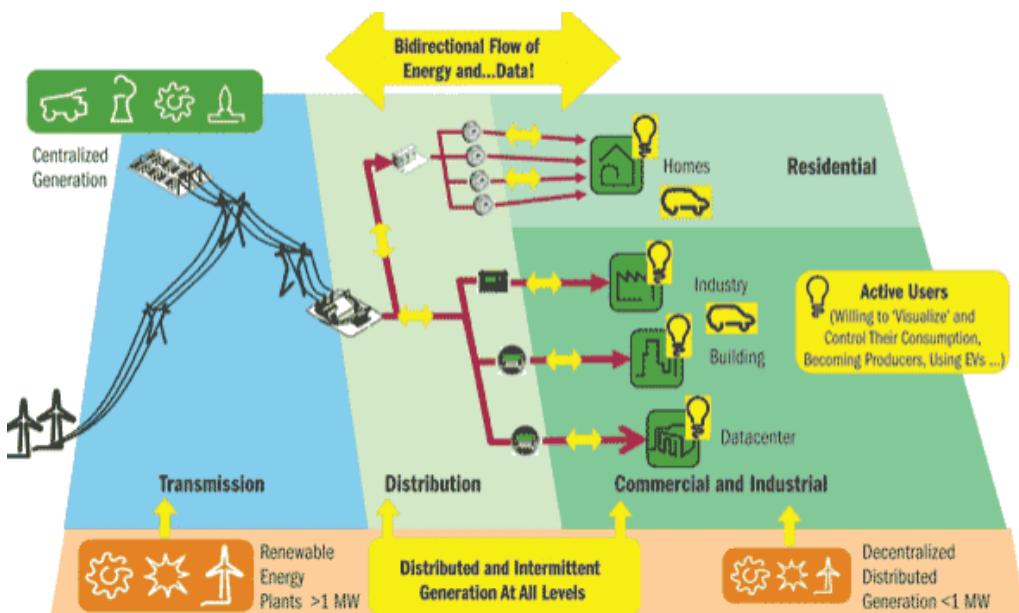


Figura 2.1: Red de distribución y flujo bidireccional de datos y energía en una *smart grid* [10]

Su base se asienta en el diseño de sistemas inteligentes coordinados capaces de obtener tanto la información respectiva a la demanda o requisitos energéticos de cada zona como la de la disponibilidad de recursos a partir de las diferentes fuentes de producción existentes. Todo ello conduce al desarrollo de un plan estratégico de distribución energética para poder conectar a todas las entidades participantes en la red entre sí.

Además, en cuanto al factor medioambiental, las SGs son un concepto clave en la transición energética y en el proceso de descarbonización. La aplicación de una distribución energética inteligente y la posibilidad de integrar al sistema fuentes renovables y limpias potencian el desarrollo de ciudades sostenibles y permiten una gran reducción de emisiones de CO₂. [1] [11]

2.1.1. Flujo energético y gestión bidireccional

Cabe destacar que una de las principales características diferenciadoras de las SGs respecto a las redes energéticas convencionales es que se fundamentan en una gestión bidireccional. Este factor determinante se puede apreciar en la Figura 2.1 y se traduce en una producción y una distribución dinámica y personalizada hacia cada uno de los usuarios de la red.

Es decir, en una red tradicional al seguir un modelo unidireccional [12], se provee energía desde el distribuidor hacia el consumidor sin llevar a cabo ningún análisis estadístico sobre el consumo que se está produciendo en las líneas finales en un determinado instante temporal. En cambio, en el contexto de las SGs, se pone el foco en las acciones de los usuarios y en la consecuente asignación de patrones de consumo eléctrico que permita predecir el comportamiento futuro de los mismos. Sin embargo, como se expondrá más adelante en la Sección 2.1.3.3, este proceso de clasificación de usuarios requerirá del análisis de grandes volúmenes de datos adquiridos a tiempo real, por lo que se aumentará la complejidad del sistema a costa de alcanzar la eficiencia.

Bajo la premisa de la característica de red bidireccional, se puede expresar que los propios usuarios son el gran pilar sobre el que se cimienta el sistema. En comparación a la red eléctrica tradicional, toman un papel mucho más activo monitorizando continuamente su comportamiento eléctrico y recopilando información para trasladarla al resto de la red. Esto tiene como ventaja una gran reducción de los costes derivados de la distribución y transmisión en el sistema, ya que se evita proporcionar más cantidad de energía de la requerida por las líneas finales [9].

Es por ello que se emplea una Infraestructura Avanzada de Medición (del inglés *Advanced Metering Infrastructure* (AMI)) con una interfaz constituida por varias capas o niveles como se puede apreciar en la Figura 2.2. De esta forma, se dividen los flujos de comunicación entre las diferentes entidades que componen la red para permitir una mejor gestión de los mismos [8].

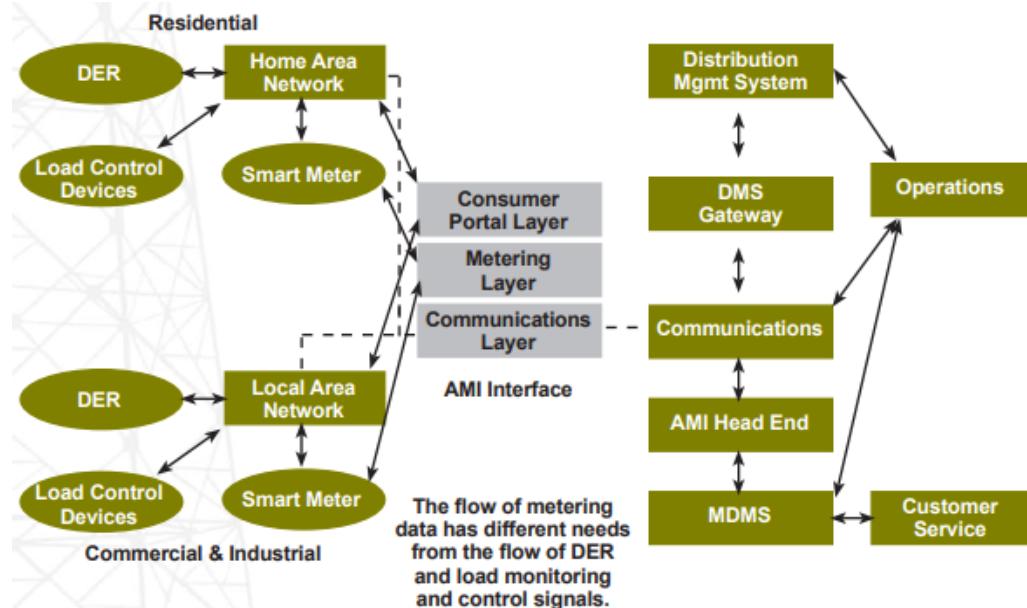


Figura 2.2: Infraestructura de una interfaz AMI para el flujo bidireccional de energía y datos [8]

2.1.2. Prosumidores y microgrids

Dentro del presente contexto, en referencia a los usuarios, es preciso introducir el término de prosumidor [13]. Se puede determinar como prosumidora a toda entidad o usuario final que tiene la capacidad de producir de forma alternativa recursos energéticos, además de poder consumirlos. En la Figura 2.3, se puede visualizar cómo se estructura la SG en un conjunto de distintas comunidades o agregaciones de prosumidores con el fin de crear microrredes de distribución y almacenamiento energético (del inglés *microgrids*).

En este caso es imprescindible que los equipos y dispositivos dedicados a la generación de electricidad se encuentren dentro de la microrred o en una ubicación cercana a la misma. Así, se podrá abastecer a los usuarios del conjunto de forma eficiente, reduciendo los costes relacionados con la transmisión energética. Este tipo de esquema puede ser aplicado en diferentes entornos, como pueden ser los residenciales, industriales o agrícolas, para conseguir descentralizar la gestión de los recursos.

Sin embargo, debe existir cierta gestión externa y coordinada de todas estas microrredes o comunidades de usuarios. Esta es llevada a cabo por una entidad denominada como operador del sistema distribuido (del inglés *Distribution System Operator* (DSO)) [13] [14]. Algunas de sus funciones principales se basan en asegurar el abastecimiento de recursos en el interior de las microrredes en función de la demanda existente y en supervisar el estado operativo la infraestructura de red para garantizar su estabilidad y seguridad.

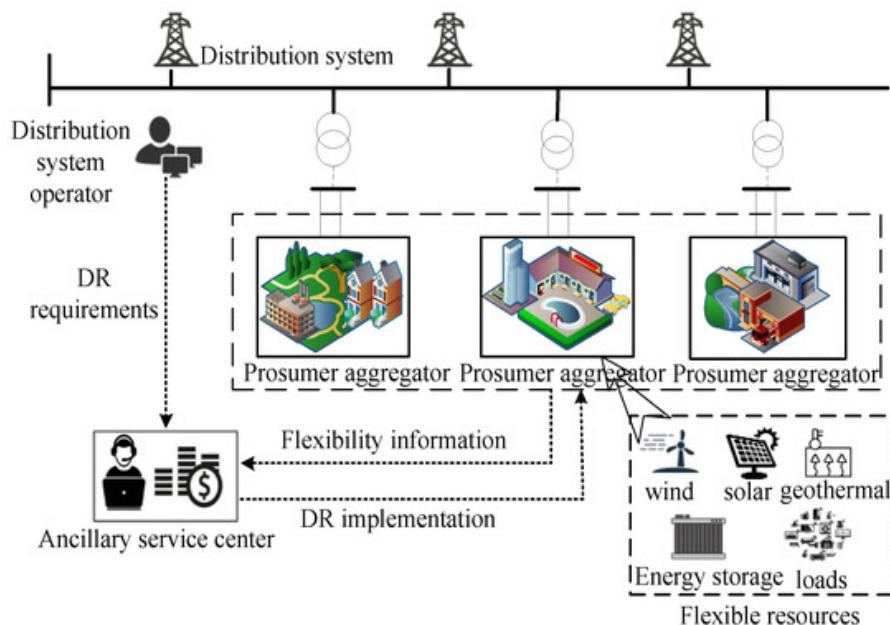


Figura 2.3: Estructura de la respuesta a la demanda en el contexto de una SG con prosumidores [13]

Por otro lado, se encuentra también la figura del centro de servicios auxiliares [13], que se encarga de recibir la información sobre los recursos energéticos de los prosumidores y cuantificar en función de la oferta y demanda el beneficio económico que obtendrán los prosumidores. Estos normalmente son gestionados por los operadores del mercado eléctrico, pero puede variar según la región o el país.

2.1.2.1. Mercado energético

En la Figura 2.4 se ilustra la estructura del modelo que presenta el mercado energético. Por un lado, las agregaciones de prosumidores posibilitan el autoconsumo y la compartición de los recursos generados por sí mismos con los demás participantes de la microrred, además de con el resto de la SG. La mayoría de los prosumidores, sobre todo en entornos

residenciales son de menor tamaño y tienen mayores complicaciones para participar en los mercados eléctricos por sí mismos.

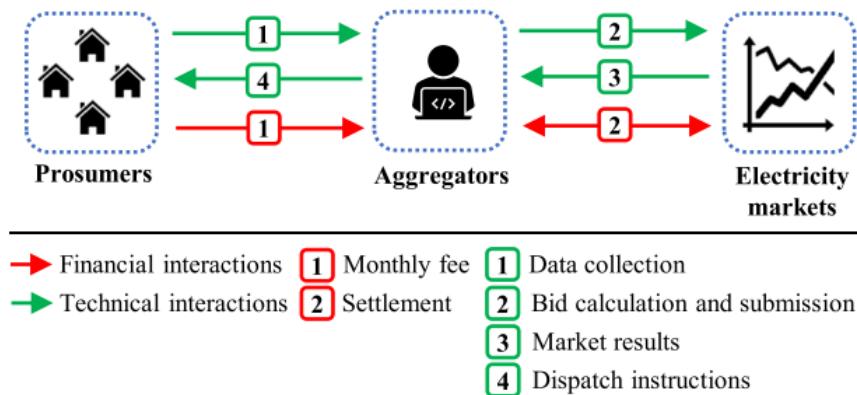


Figura 2.4: Modelo basado en las interacciones de los agregadores con los prosumidores y el mercado eléctrico [15]

Por ello, los agregadores [16] [13] se encargan de facilitar esta tarea para que una cantidad de prosumidores locales puedan actuar en conjunto como fijadores de precios de compra/venta en el mercado eléctrico global. Es decir, cada agregador se encarga de representar a un conjunto determinado de prosumidores y opera como entidad intermediaria entre los mismos y el conjunto de mercados energéticos. Cada prosumidor debe pagar a cambio un plan mensual a la compañía para cubrir los costes eléctricos y así, dejar la responsabilidad de las transacciones a los agregadores para que operen de forma flexible.

Por otro lado, en esta operativa también incide el DSO, el cual se encarga de que las transacciones de compra/venta sigan el cumplimiento de los estándares y regulaciones establecidos en el mercado energético. A partir del funcionamiento expuesto y de la Figura 2.4, se pueden diferenciar los tipos de interacciones que se producen en este modelo [15]:

- Interacciones técnicas
 1. Recopilación y adquisición de datos sobre los recursos de los prosumidores para su posterior procesamiento en los agregadores.
 2. Cuantificación y presentación de ofertas desde los agregadores hacia los mercados mayoristas de electricidad.
 3. Comunicación de los resultados de diferentes mercados a los agregadores.
 4. Envío de instrucciones o recursos distribuidos en función de los resultados del mercado.

- Interacciones financieras
 1. Pago de tarifa mensual de cada prosumidor al agregador.
 2. Liquidación de las transacciones trasladadas desde los prosumidores en el mercado eléctrico (cobro por demanda y pago por generación).

En cuanto al proceso de fijación de precios, generalmente, se adoptan tarifas minoristas dinámicas según el instante temporal, como pueden ser la fijación de precios por tiempo de uso (del inglés *Time-of-Use Pricing* (TOU)) o en tiempo real (del inglés *Real Time Pricing* (RTP)). Dentro de este contexto es preciso introducir los programas de gestión del lado de la demanda (del inglés *Demand Side Management* (DSM)).

Como término, un programa DSM [17] es un programa basado en el control de las interacciones de consumo y de gestión de cargas residenciales desde el lado del cliente. Considerando este modelo de interacciones, se puede expresar que un DSM tiene como pilar fundamental la distribución de recursos energéticos de manera eficiente y acorde a la demanda y a la oferta. Permite reducir el coste de la adquisición energética y los costes asociados a la distribución como consecuencia de minimizar el número de interacciones necesarias entre el prosumidor y el resto de entidades del sistema.

Por lo tanto, volviendo a los modos de dinamización de las tarifas, si por ejemplo se emplea RTP, se buscará lograr el equilibrio de la demanda en tiempo real, modificando las cargas de los consumidores en las horas pico [18]. Es decir, cada uno de los usuarios actuará individualmente en función de los precios dinámicos en el tiempo, comunicándose directamente con la compañía energética como se muestra en la Figura 2.5.

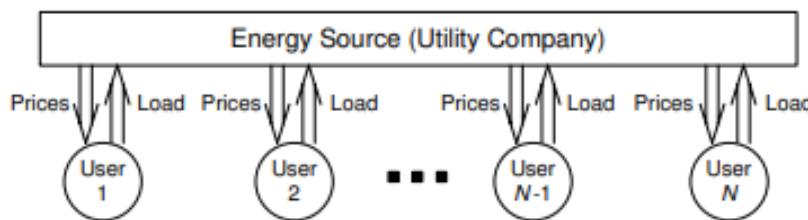


Figura 2.5: Estrategia de DSM basada en interacciones individuales de cada consumidor con la compañía [19]

Con este proceso, como el consumidor puede saber en tiempo real la tarifa a la que está consumiendo energía, puede trasladar su propia carga desde las horas donde el precio es más alto (horas pico) a las de precio menor (horas valle). Esta gestión produce por tanto, menores picos de demanda y contribuye a la bajada de los precios [17] [19].

No obstante, el diseño de un programa DSM ideal en el contexto de las SGs debería permitir también las interacciones entre los mismos prosumidores dentro de una zona residencial o microgrid. Estas interacciones generalmente se automatizan a través de equipos dedicados a una comunicación digital bidireccional. Como se aprecia en la Figura 2.6, este proceso tiene como fin conducir a una coordinación de las acciones respectivas a las cargas de un área determinada. Por tanto, en este caso la monitorización de las cargas totales para todos los nodos participantes en un instante determinado aportará la información necesaria sobre la relación entre la potencia pico y promedio (del inglés *Peak-to-Average Ratio* (PAR)) y contribuirá a la fijación del precio unitario para ese mismo instante [19].

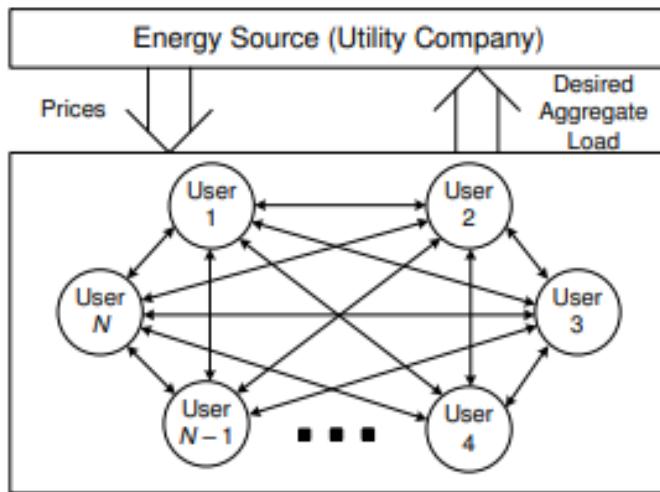


Figura 2.6: Estrategia de DSM para SGs basada en interacciones entre los usuarios y la compañía [19]

2.1.3. Estructura de una Smart Grid

Una SG está constituida por múltiples elementos diferentes como se puede visualizar en la Figura 2.7. Cada uno de ellos está dedicado a uno de los procesos principales, que se pueden dividir en generación, distribución y consumo [6].

2.1.3.1. Generación

Como ya se introducía en la Sección 2.1.2, dentro de una SG, el prosumidor se trata de la figura que potencia principalmente el uso de energías renovables, las cuales pueden ser obtenidas por ejemplo a partir de generadores térmicos fotovoltaicos (del inglés *Photovoltaic Thermal* (PVT)) o de turbinas eólicas. Para posibilitar posteriormente el uso



Figura 2.7: Estructura de componentes de una SG [6]

de la energía generada, esta debe ser convertida y acondicionada mediante dispositivos dedicados, como pueden ser las unidades combinadas de calor (del inglés *Combined Heat and Power* (CHP)) o las de conversión a gas (del inglés *Power-to-Gas* (P2G)) [13].

Las unidades CHP [20] [21], como se puede apreciar en la Figura 2.8, llevan a cabo un aprovechamiento del calor residual que deriva del proceso de generación de electricidad. Se emplean como respaldo eléctrico, ya que se componen de sistemas de almacenamiento de energía térmica (del inglés *Thermal Energy Storages* (TES)) para poder separar la producción y el uso del calor y la energía. Luego, este calor almacenado se puede emplear en aplicaciones térmicas como la calefacción o en otros procesos enfocados a entornos industriales.

Es por ello que las unidades CHP mejoran la eficiencia del sistema energético, contribuyendo a un uso mucho más eficaz de los recursos y a una reducción de las pérdidas que se producen en la transmisión. Además del TES, utilizado para almacenar el calor producido, una unidad CHP también se constituye de un motor acoplado a un generador eléctrico para llevar a cabo el proceso de generación de electricidad y calor. Como se puede apreciar en la Figura 2.9, todas las operaciones de la unidad y las interacciones que se producen entre módulos se coordinan desde el gestor de operaciones para que se produzca un correcto funcionamiento.

COMBINED HEAT & POWER

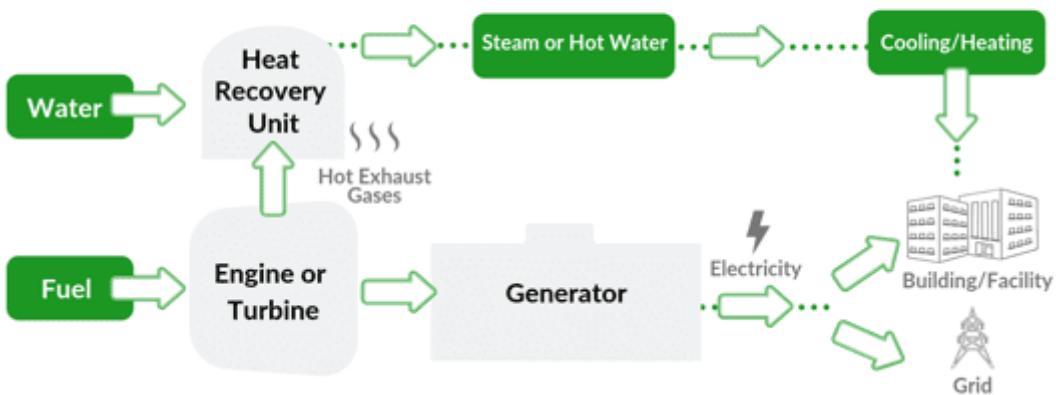


Figura 2.8: Esquema de funcionamiento de una unidad CHP [20]

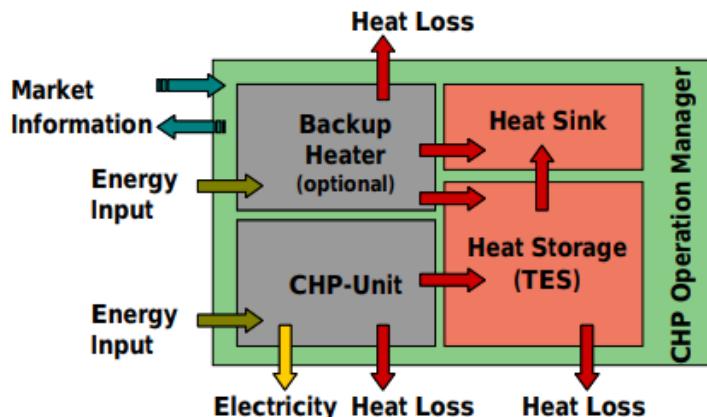


Figura 2.9: Gestión de operaciones en una unidad CHP [21]

Por otro lado, la unidad P2G, mencionada anteriormente, se encarga de la conversión de la electricidad en gases sintéticos, como son el hidrógeno o el metano. Es de gran importancia, ya que este tipo de gases son más fácil de transportar y almacenar que la electricidad, por lo que se simplifica la gestión energética. Como se puede apreciar en la Figura 2.10, el objetivo del P2G se basa en descomponer el agua mediante un proceso de electrólisis a partir de electricidad proveniente de fuentes renovables. Después, el hidrógeno que se produce se puede utilizar directamente o procesarlo de forma adicional en metano [13].

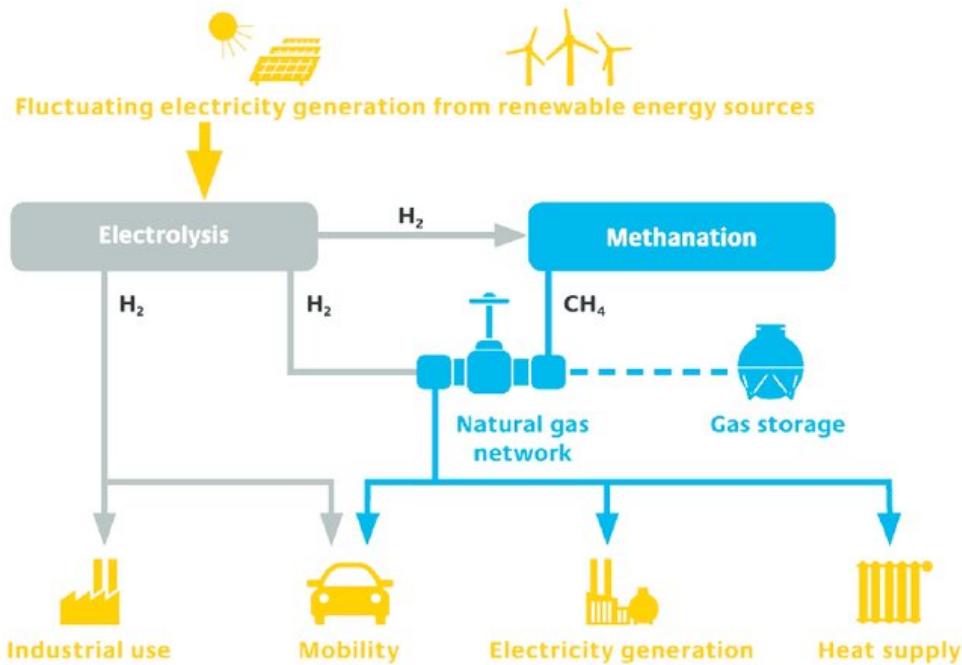


Figura 2.10: Representación del proceso P2G [22]

Dentro del contexto de la generación de energía también es importante destacar algunos equipos, como son los sistemas de control y adquisición de datos (del inglés *Supervisory Control and Data Acquisition* (SCADA)) [23]. Estos pueden ser instalados en generadores, como paneles solares fotovoltaicos o turbinas eólicas, consiguiendo una monitorización remota de los mismos. Mediante la información recopilada a tiempo real permiten conocer los niveles de generación y establecer una predicción de la disponibilidad energética que habrá en el sistema o en un área determinada del mismo.

2.1.3.2. Distribución

Un dispositivo importante en el campo de la distribución energética es la Unidad de Medición de Fasores (del inglés *Phasor Measurement Unit* (PMU)) [24]. Se emplea para medir con una alta precisión los fasores de tensión y corriente en la red eléctrica, proporcionando información relevante sobre las magnitudes y fases de las ondas sinusoidales. También, es empleado en los generadores y debe contar con una alta tasa de muestreo para capturar eventos transitorios o cambios rápidos en la red. En otros términos, debe tener la capacidad de identificar o detectar a tiempo real posibles anomalías que se puedan dar en la distribución. Teniendo esto en cuenta, se puede expresar que se trata de un componente imprescindible para comprobar y garantizar la estabilidad de la SG.

La confiabilidad y la seguridad de la red también reside en los Sistemas de Transmisión de Corriente Alterna Flexibles (del inglés *Flexible AC Transmission System (FACTS)*) [25] [26]. Estos vienen dados por la necesidad de superar las limitaciones técnicas introducidas por las redes eléctricas, como son las térmicas o las respectivas al voltaje. En otros términos, incrementan la potencia transmitida y aportan flexibilidad al permitir modificar de forma dinámica los parámetros eléctricos ante cambios en la configuración de la red.

Los FACTS [27] [26] incluyen todos los elementos electrónicos basados en tecnología de alta potencia y que son empleados dentro de una SG para la transferencia de energía de CA y el control de la potencia reactiva. También, realizan tareas de reducción de impedancia en las líneas de transmisión y de optimización del factor de potencia y pueden actuar tanto a nivel individual como de forma coordinada con otros controladores. Los FACTS se dividen en dos generaciones: la primera emplea interruptores controlados por tiristores y la segunda tiene como base convertidores estáticos de commutación. En la Tabla 2.1 se visualizan las tecnologías FACTS más relevantes.

Poniendo el enfoque en la garantía de estabilidad de una SG, uno de los sistemas más importantes de los expuestos en la Tabla 2.1 es el *Static Var Compensator (SVC)* [25]. En la Figura 2.11 se puede apreciar un ejemplo de instalación de compensador, ubicado en el municipio noruego Sylling y conectado a un sistema de 420 kV.



Figura 2.11: Instalación de un SVC en Sylling (Noruega) [25]

Generación	Tipo	Descripción
1º	<i>Thyristor-Controlled Series Capacitor</i> (TCSC)	Controla el flujo de potencia tanto reactiva como activa por la línea de transmisión mediante el ajuste de impedancia en serie y amortigua las oscilaciones.
	SVC	Absorbe o suministra potencia reactiva según las necesidades de una línea de transmisión a través de la variación de la susceptancia en paralelo. Ayuda a mantener el voltaje estable en la red y proveen un aumento de la capacidad de transferencia de energía.
2º	<i>Static Synchronous Compensator</i> (STATCOM)	Compensa la potencia reactiva al igual que el SVC, pero en este caso empleando electrónica de potencia para proporcionar una respuesta más rápida.
	<i>Unified Power Flow Controller</i> (UPFC)	Combina las funciones de un TCSC y un SVC, teniendo la capacidad de controlar en una línea de transmisión tanto la impedancia en serie como la susceptancia en paralelo.
	<i>Static Synchronous Series Compensator</i> (SSSC)	Modifica dinámicamente la impedancia y es capaz de controlar la fase y la amplitud de la tensión en la línea.
	<i>Interline Power Flow Controller</i> (IPFC)	Conecta varias líneas de transmisión en paralelo. No obstante, modula la impedancia y la fase de la tensión de cada línea de transmisión de forma independiente, lo que permite operar sobre el flujo de potencia de una forma controlada. Mejora la capacidad de transmisión y reduce las pérdidas energéticas.

Tabla 2.1: Tecnologías FACTS de 1^a y 2^a generación

Los compensadores estáticos son capaces de detectar grandes caídas de tensión resultantes a la producción de un cortocircuito o a la pérdida de líneas de transición en un área de la red. Esto es importante, ya que una detección rápida de una avería permite restaurar en un corto período de tiempo la tensión del área afectada sin que el problema escale a otras partes de la red. En otros términos, aísla este área del resto de la red eléctrica. Además, el SVC asegura que el proceso de restauración se produzca paulatinamente para que los efectos producidos por el cortocircuito sean prácticamente imperceptibles en los puntos de carga del área afectada.

Siguiendo en el ámbito de la distribución energética, es importante también exponer las tecnologías enfocadas a la transmisión de electricidad a larga distancia, como son las Corriente Continua de Alto Voltaje (del inglés *High Voltage Direct Current* (HVDC)) [28]. Como su nombre indica, utilizan corriente continua para ello, lo que minimiza las pérdidas en el proceso de transmisión respecto al caso de la Corriente Alterna de Alto Voltaje (del inglés *High Voltage Alternating Current* (HVAC)), además de permitir portar mucha más potencia. Esto es fundamental cuando se pretende integrar al sistema global fuentes de energía ubicadas en áreas remotas, sobre todo dentro del contexto de las SGs. Generalmente, las fuentes de energías renovables, como pueden ser la solar o la eólica, se encuentran alejadas de los puntos geográficos con mayor demanda y se requiere una respuesta rápida ante cambios de carga para mantener la estabilidad del sistema.

Entrando en detalle, cuando se transmiten grandes cantidades de potencia a largas distancias con líneas de HVAC, se generan ángulos eléctricos entre los voltajes y las corrientes en la línea muy grandes que pueden producir oscilaciones descontroladas y por tanto, desencadenar inestabilidades que llegarían a escalar a lo largo del sistema. Las tecnologías HVDC simplifican la interconexión entre microrredes y permiten una mayor integración de sistemas dedicados al almacenamiento energético para facilitar la gestión de los recursos disponibles. Sin embargo, presentan ciertas desventajas, destacando el alto coste que supone su implementación, ya que al final únicamente se pueden emplear en el caso de aplicaciones punto a punto.

En la Figura 2.12 se muestra como ejemplo una infraestructura de transmisión de energía eólica a larga distancia. Esta energía proviene de una fuente remota y se transporta a través del medio marino para llegar hasta los usuarios. Como se puede apreciar, es necesario implementar estaciones de conversión de alto voltaje de corriente alterna a continua y viceversa en los extremos de las líneas HVDC.

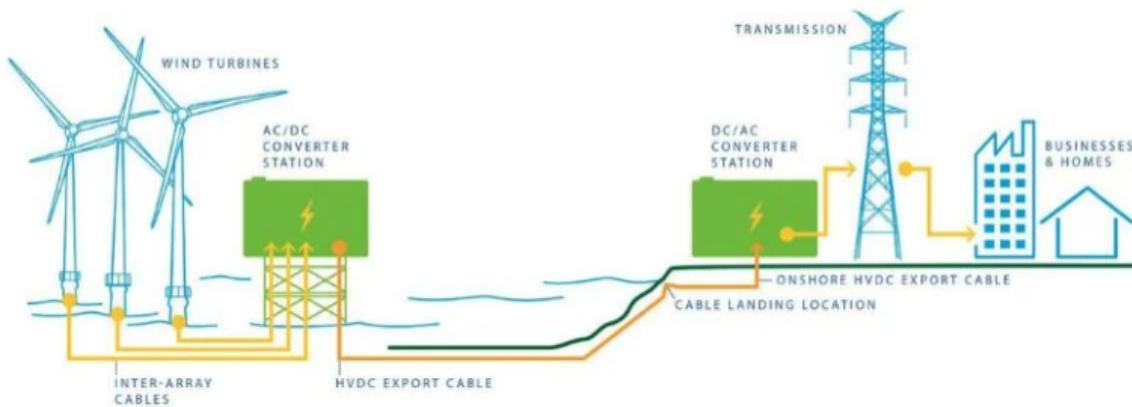


Figura 2.12: Representación de un sistema de transmisión de energía eólica con líneas HVDC [29]

2.1.3.3. Consumo

Entrando en detalle en las ubicaciones finales del sistema y por tanto, en el proceso de consumo, es imprescindible conocer la cantidad de energía demandada por los clientes que pertenecen a una SG. La instalación de sensores o medidores inteligentes (del inglés *smart meters*) [30] en las viviendas posibilitan el registro constante de datos respectivos al consumo de energía, niveles de voltaje, corriente y factor de potencia. Estos son almacenados, analizados y procesados por las distribuidoras de energía para obtener a partir de los mismos la información necesaria sobre el comportamiento de los usuarios.

Como se había introducido en el Apartado 2.1 el proceso de adquisición de datos por parte de los usuarios se trata de uno de los pilares más relevantes de cara a la optimización energética del sistema. Los objetivos principales que se pretenden con este proceso se engloban en la reducción del consumo y la optimización de la distribución y el beneficio. Respecto a este último, las compañías energéticas a partir de su base de clientes estudian las estrategias de categorización de los mismos para definir el sistema de fijación de precios dinámicos [30].

No obstante, la gestión de los sensores finales en las SGs se caracteriza por su alta complejidad, debido a los grandes volúmenes de datos a manejar. Se requiere un procesamiento eficiente a tiempo real para analizar datos capturados de múltiples fuentes con el fin de evitar latencias o bloqueos en el sistema. Es por ello que se dificulta el empleo de herramientas convencionales de gestión de bases de datos y se requiere una solución más avanzada mediante la aplicación de tecnologías enfocadas al *Big Data*. Se profundizará más sobre ello en la Sección 2.4.

2.1.4. Seguridad en Smart grids

Como se ha introducido en la Sección referente a la figura del prosumidor (ver Sección 2.1.2), el DSO tiene la capacidad de operar ante problemas y fallos en la red tomando decisiones y dando una respuesta rápida. En determinados casos, las incidencias se pueden dar por la configuración del propio sistema, si esta no se realiza de forma correcta y eficiente. No obstante, también es preciso contemplar la posibilidad de ataques externos que atenten contra el sistema de la SG.

Uno de los ataques más peligrosos en este ámbito es el de Inyección de Datos Falsos (del inglés *False Data Injection* (FDI)) [31] y, como su nombre indica, se basa en la inyección de paquetes maliciosos en la red con el fin de bloquear los servicios de la misma y conseguir la autorización para realizar operaciones restringidas. El proceso se puede realizar actuando directamente a través de los sensores finales (*smart meters*) o secuestrando el canal de comunicación. Algunas técnicas empleadas por los atacantes son las siguientes [31]:

- **Fallo del dispositivo:** Se aplica un ataque de Denegación de Servicio (del inglés *Denial of Service* (DoS)) a los sensores. Cabe destacar que generalmente los sensores que se encuentran en las ubicaciones finales tienen una capacidad limitada de conexión, que derivaría en grandes latencias en la comunicación de datos con el resto de entidades de la red. En este momento, el atacante se encarga de suplantar al host en cuestión para enviar los paquetes falsos en su nombre.
- **Cracking:** Se descifran las contraseñas para obtener acceso a los equipos mediante ingeniería social o fuerza bruta. De la misma forma que el anterior, la limitación de los recursos computacionales de los sensores produce la inexistencia de mecanismos de contraseñas lo suficientemente seguros. Además, la mayoría de ellos emplean como protocolos de comunicación Modbus/TCP o DNP 3.0/TCP, los cuales implementan una transmisión de texto en formato plano y sin cifrado. Por ello, un atacante puede tener la posibilidad de monitorear y capturar el tráfico si consigue acceso.
- **Envenenamiento de tablas Address Resolution Protocol (ARP)** (del inglés *ARP Spoofing*) [32]: Consiste en enviar mensajes falsos ARP para possibilitar la captura y la alteración de los paquetes que se están transmitiendo por la red, a través de un ataque *Man In The Middle* (MITM).

Teniendo esto en cuenta, es imprescindible definir los mecanismos de detección y mitigación a emplear para evitar que se produzcan ataques FDI [31]:

- **Mecanismo de autenticación estrictos:** Los datos que se suministran al DSO o a otras entidades de control de la SG deben de ser autenticados para comprobar su

procedencia. Para ello, se emplean por ejemplo, marcas de tiempo para evitar ataques de repetición y protocolos de seguridad como *Transport Layer Security Protocol* (TLS) o *Secure Sockets Layer* (SSL), además de *Secure Hash Algorithm* (SHA) y *Hash Message Authentication Code* (HMAC).

- **Gestión dinámica de claves:** Se actualizan las claves con una determinada frecuencia para incrementar la seguridad de los estándares IEEE 802.11s y evitar ataques DoS.
- **Empleo del protocolo *Secure Neighbor Discovery* (SEND)** [33]: Para evitar los ataques de *ARP Spoofing* se emplean pares de claves *Rivest, Shamir y Adleman* (RSA) para poder garantizar en el proceso de enrutamiento que los mensajes que tienen como origen un determinado host pertenecen al mismo y evitar ataques MITM.

2.1.5. Desafíos futuros de las smart grids

Las SGs, a pesar del progreso tecnológico que han proporcionado en el ámbito de la gestión energética en la última década, no están exentas de desafíos importantes que deben de abordarse para seguir potenciando su crecimiento a futuro. La segunda generación de SGs viene dada por la necesidad de implementar las siguientes mejoras: [6] [34]

- **Inteligencia mejorada:** Será preciso poner el foco en el desarrollo de nuevas infraestructuras avanzadas de medición inteligente para afrontar cada vez una mayor demanda y globalizar el mercado energético.
- **Fortalecimiento de la red:** Se debe garantizar una suficiente capacidad de transmisión para favorecer la transmisión de larga distancia y el empleo de fuentes energéticas aisladas, como pueden ser los parques eólicos en el mar.
- **Adaptación de la red al empleo de vehículos eléctricos:** Se requiere preparar la arquitectura de las SGs para el creciente uso de vehículos eléctricos que se espera en la próxima década.
- **Gestión de la generación intermitente y del almacenamiento:** Es importante potenciar el empleo de los recursos de generación a pequeña escala, sobre todo en el caso de los entornos residenciales, para conseguir cada vez una mayor descentralización de la red. De la misma manera, se deben abordar las limitaciones actuales que presentan los sistemas de almacenamiento energético para permitir una mayor acumulación de recursos provenientes de fuentes de generación intermitente, como

es la solar y la eólica. Esto es imprescindible para garantizar la disponibilidad energética y poder afrontar los posibles picos de demanda que se produzcan a lo largo del tiempo.

- **Regulación favorable e incentivos económicos:** Se puede potenciar la transformación de las redes energéticas mediante el establecimiento de políticas regulatorias que fomenten la inversión [35] en SGs y mediante colaboraciones entre el sector público y el privado. De la misma forma, implementar estructuras de tarifas de precios dinámicos motivará a un número mayor de usuarios finales a adoptar tecnologías inteligentes para obtener beneficios económicos.

2.2. IoT

El Internet de las Cosas (del inglés *Internet of Things* (IoT)) [36] [37] se define como la integración de múltiples sensores o dispositivos físicos interconectados entre sí que se comunican a través de una red inalámbrica y recopilan información en tiempo real.

La evolución del *Internet of Things* (IoT) tal y como se conoce en la actualidad tiene sus inicios en los años 90. Sin embargo, en ese entonces estas tecnologías se enfrentaban con la problemática del gran tamaño que presentaban los chips, por lo que se produjo un avance lento de las mismas. A medida que se permitió la reducción del volumen los dispositivos electrónicos y se mejoró la eficiencia y la capacidad de computación de los chips, se facilitó el progreso del IoT. En los últimos años, la integración de la tecnología 5G o de quinta generación móvil, ha supuesto el incremento de la velocidad de transmisión, procesamiento y análisis de los datos en los sistemas IoT. La capacidad de administrar a gran escala multitud de dispositivos físicos y la próxima implementación del 6G, depara un futuro en el que las tecnologías IoT estarán presentes en numerosos ámbitos y sectores.

Una de las grandes ventajas que proporcionan los sistemas IoT es la aparición de los entornos *Machine To Machine* (M2M) [38], en los cuales no es necesaria la intervención humana para posibilitar la transmisión y recepción de datos entre los elementos de la red. También, cabe destacar la reducción del coste de computación, la automatización de las tareas y la monitorización en tiempo real. Sin embargo, en cuanto a posibles desventajas, pueden presentar ciertas vulnerabilidades en cuestiones de privacidad y seguridad. Por ello, en la construcción de un sistema IoT se deben centrar esfuerzos en blindar los distintos elementos que lo componen frente a posibles ataques que puedan extraer información delicada o sabotear el funcionamiento. En cuanto a la estructura de un sistema IoT, se pueden diferenciar cuatro componentes que contribuyen al funcionamiento del mismo [36] [37]:

- **Dispositivos inteligentes:** Se trata de cualquier elemento físico al que se le pueda asignar una dirección IPv6 y que tenga como mínimo capacidad de computación para recopilar datos del entorno o del usuario y transmitirlos a través de la red.
- **Conectividad:** En este campo entran varias posibilidades de conexión: Wi-Fi, Bluetooth, redes de área amplia y baja potencia (del inglés *Low Power Wide Area Networks* (LPWAN)) como LoRaWAN, entre otras.
- **Aplicación:** Se compone de un conjunto de servicios de nube que se encargan de integrar y procesar todos los datos recopilados por los dispositivos del sistema IoT. Además, en algunos ámbitos, emplea técnicas de ML para analizar la información que recibe y optimizar la toma de decisiones en el sistema.
- **Interfaz gráfica:** Desde la misma, el usuario puede gestionar y controlar el conjunto de elementos de la red. La interfaz está conectada directamente a la nube y puede tratarse de una aplicación móvil o web.

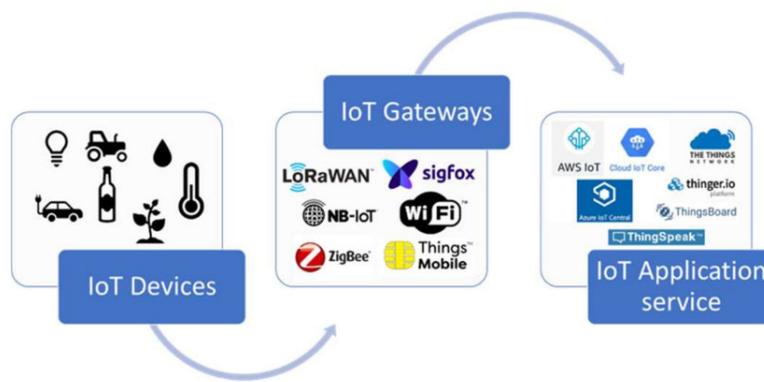


Figura 2.13: Infraestructura de componentes de un sistema IoT [39]

2.2.1. IoE

Considerando el contexto energético en el que se engloba este TFM, y entrando con más detalle en el marco de las tecnologías IoT, es preciso destacar dentro de las mismas la subcategoría denominada como el Internet de la Energía (del inglés *Internet of Energy* (IoE)) [34]. Como concepto, el IoE comprende todo el paradigma de operación de los elementos que constituyen una red energética. Es decir, lleva a cabo la integración de todos los dispositivos, sensores o equipos informáticos en una misma estructura, la cual está basada en internet para poder monitorizar y controlar remotamente el estado de cada punto de la red.

Se puede expresar que, con la implementación de tecnologías basadas en IoE, se buscan objetivos muy similares a las SGs. No obstante, el IoE va más allá y supone un control y una gestión de la red a mayor escala. En otros términos, no solo se constituye por los elementos de la propia infraestructura eléctrica, sino que también permite una gestión energética a nivel interna de los hogares, mediante la inclusión de electrodomésticos y otros dispositivos electrónicos.

2.3. DEN2NE

Distributed ENergy ENVironments and Networks (DEN2NE)¹ [3] [40] se presenta como un algoritmo diseñado por el equipo de investigación NetIS de la UAH para la automatización del proceso de encaminamiento y de compartición de recursos de forma eficiente. Su aplicación en este TFM se enfoca en el ámbito de las SGs para llevar a cabo el proceso de distribución energética entre los diferentes prosumidores (ver Sección 2.1.2) que participan en la red. No obstante, DEN2NE está diseñado para emplearse en cualquier entorno distribuido donde los nodos necesiten colaborar, compartir e intercambiar recursos para lograr objetivos comunes.

El algoritmo se enfoca en la búsqueda de nodos vecinos para llevar a cabo la distribución energética. En este proceso de intercambio de recursos se asumen conexiones bidireccionales entre los nodos para representar la oferta y la demanda existente. Es preciso indicar que DEN2NE se fundamenta en los siguientes principios: escalabilidad, versatilidad, flexibilidad y resiliencia. En el caso de la escalabilidad, el algoritmo minimiza los tiempos de convergencia para posibilitar la aplicación del mismo en topologías densas, tomando importancia al consumo de recursos de memoria y minimizando el número de mensajes transmitidos.

En cuanto a la resiliencia, se obtienen rutas de espaldo de cada prosumidor a un nodo raíz o *gateway*, que, en el caso de uso de las SGs, daría acceso a la red de distribución eléctrica y se asumiría que está conectado a recursos infinitos. Este nodo toma la responsabilidad de funcionar como balanceador, puesto que el principal objetivo de DEN2NE viene dado por la compensación de los recursos de la red a nivel global. A modo de visualizar de una forma gráfica este proceso, se muestra en la Figura 2.14 un ejemplo de una red compuesta por seis nodos y un *gateway*. Como se puede apreciar, se representa tanto la conectividad que existe entre los nodos (color negro), como el sentido del intercambio de los recursos hacia el *gateway* (color naranja).

¹<https://github.com/NETSERV-UAH/den2ne-Alg>

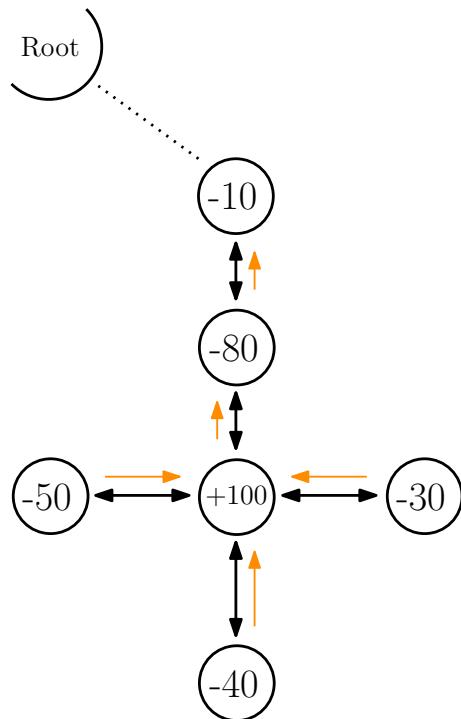


Figura 2.14: Ejemplo del proceso de balance de recursos [3]

Teniendo esto en cuenta, el primer paso que lleva a cabo DEN2NE se basa en una asignación jerárquica de etiquetas. Mediante este proceso, cada nodo obtendrá uno o varios identificadores numéricos (IDs), a partir de los que podrá definir su posición relativa en la topología respecto al nodo raíz. Al aplicar este paso, se calculan todos los caminos que existen desde cada nodo hacia el *gateway* y se tratan los posibles bucles que puedan producirse al asignar las etiquetas como se representa en la Figura 2.15.

En el caso de que DEN2NE sea configurado para asignar varias etiquetas a cada nodo, se añade un paso de selección, en el que se escogerá una única etiqueta de las asignadas anteriormente. Es decir, para cada nodo se comprobará qué identificador (ID) es el más apropiado y se activará, tomando en consideración alguno de los seis criterios establecidos por el algoritmo. Estos criterios vienen determinados por el número de saltos o distancia (física o lógica) al nodo raíz, las pérdidas de enlace, el balance de potencia o las combinaciones de algunos de los anteriores. Con el proceso de selección, se establecerá la topología lógica y, por tanto, los caminos finales desde cada nodo al *gateway*. No obstante, las etiquetas no seleccionadas pueden ser útiles ante la necesidad de caminos opcionales, en el caso de que alguno de los nodos falle.

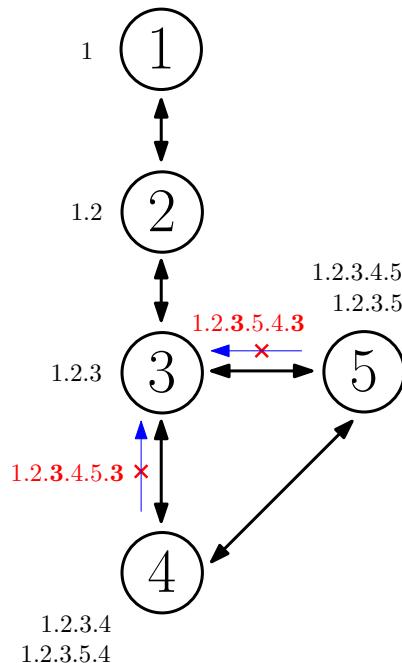


Figura 2.15: Proceso de asignación de etiquetas [3]

Como se introducía al comienzo de la presente Sección, el objetivo del algoritmo es alcanzar una situación final de balance global energético en la red. Para que este proceso sea eficiente, se debe seguir un orden de distribución basado en la longitud del identificador que caracteriza a cada nodo. DEN2NE, al ser centralizado, puede recolectar la lista de etiquetas de la topología completa y comenzar el proceso por los nodos más lejanos al *gateway*.

Por consiguiente, cada uno de los nodos existentes redireccionará el propio exceso de demanda u oferta de energía hacia su nodo padre, repitiéndose el procedimiento hasta llegar al nodo raíz. De forma concluyente, se puede apreciar en la Figura 2.16 cómo se definirá un escenario de red en el que todos los nodos tendrán un valor energético nulo, a excepción del nodo raíz, que acumulará el valor total de todos los intercambios producidos en la red. Cabe destacar que en el proceso de distribución energética es configurable el tipo de escenario, que puede ser ideal, con pérdidas, con capacidad de enlace limitada o con pérdidas y capacidad de enlace limitada. De la misma forma, el algoritmo establece para cada nodo los recursos energéticos iniciales, siguiendo una función de densidad de probabilidad.



Figura 2.16: Proceso de distribución energética [3]

2.4. Big Data

El *Big Data* [30] se puede definir como el manejo y el análisis de un conjunto extremadamente grande de datos, los cuales provienen de fuentes diferentes y no correlacionadas. Debido a su complejidad y volumen, no pueden ser procesados con software o herramientas tradicionales de gestión de bases de datos y se requieren soluciones tecnológicas más avanzadas. El concepto de *Big Data* se basa en tres componentes principales, denominadas como las 5 Vs [41] [42]:

- **Variedad:** Hace referencia a la heterogeneidad de la información. Existen distintos tipos de datos a procesar de diferentes fuentes, formas y resoluciones. Se puede simplificar su gestión si estos son estructurados (en forma de tablas de bases de datos) o dificultar, si estos no tienen una estructura definida (en formato de texto o imagen). También, pueden ser semiestructurados (en formato JSON o XML).

- **Velocidad:** Es un parámetro crítico, ya que algunos entornos requieren de una toma de decisiones a tiempo real y los datos deben de ser generados, procesados y analizados rápidamente.
- **Volumen:** Es imprescindible tener la capacidad de gestionar una gran cantidad de datos (en el orden de los TB) que además, es creciente en el tiempo. El almacenamiento y la minería de datos deben de ser eficientes para manejarlos de forma correcta
- **Veracidad:** Mide la calidad y confiabilidad de la información que se adquiere. Se comprueba su autenticidad e integridad para asegurar que proviene de una fuente legítima, por lo que se deben aplicar mecanismos de seguridad como la encriptación o herramientas de detección y mitigación de ataques a la red (ver Sección 2.1.4). También, se verifica que la información no contenga errores y que sea lo más precisa posible.
- **Valor:** La masividad de datos introduce mucho ruido y confusión en la información y debe de seleccionarse solamente aquella de utilidad, aplicando minería de datos y procesamiento adicional.

Durante los últimos años el *Big Data* se ha adoptado en multitud de sectores y campos como en las telecomunicaciones, las finanzas, el comercio, la medicina, el transporte o la investigación científica. Como se ha expuesto en los apartados anteriores (ver Sección 2.1 y, en particular, 2.1.3.3) y, poniendo enfoque en los objetivos de este TFM (ver Sección 1.2), el *Big Data* tiene un gran potencial en el ámbito de las SGs.

En este contexto, la efectividad del empleo de la información reside en la búsqueda de la correlación entre los datos adquiridos de la red eléctrica y el resto de características que pueden influir en los mismos, como son el comportamiento de los usuarios finales, las condiciones de estabilidad, la disponibilidad de recursos, el estado de las cargas o las condiciones climáticas en un determinado instante temporal. La coordinación de las mediciones que se realizan en la red junto con la información que se dispone del entorno es crucial para la monitorización, el control y la operativa llevada a cabo dentro de la SG [30].

En otros términos, un análisis efectivo del *Big Data* conducirá a una buena toma de decisiones en el sistema y, en consecuencia, a su optimización. Este análisis se constituye del empleo de herramientas dedicadas al procesamiento distribuido, el almacenamiento en la nube, la minería de datos y a técnicas y algoritmos de aprendizaje automático (ML).

2.4.1. Computación en nube

En la Figura 2.17 se representa la infraestructura de procesos dedicados al *Big Data* en un ámbito de SGs. Como se puede visualizar, se compone de tres capas operativas: la superior está enfocada al almacenamiento y computación de datos, la intermedia, a la gestión, compartición e integración de datos de diferentes aplicaciones o fuentes y la inferior, a todos los procesos y técnicas que se encargan del procesamiento, minería, clusterización y clasificación final [30].

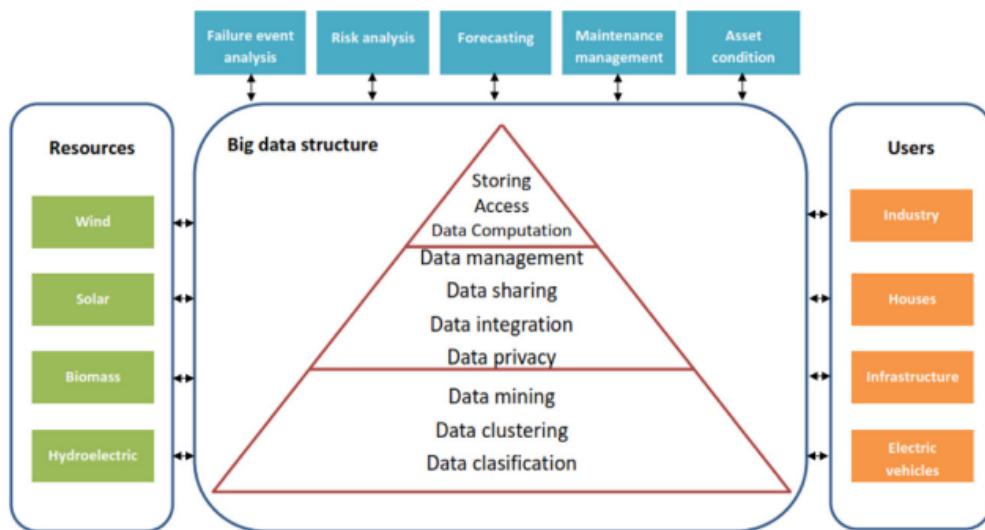


Figura 2.17: Infraestructura de *Big Data* enfocada al ámbito de las SGs [30]

Para implementar esta infraestructura, es preciso hacer uso de un entorno de computación en nube. Dentro de este contexto, las grandes tecnológicas como Google, Amazon y Microsoft han puesto sus esfuerzos en los últimos años en desarrollar y optimizar sus propios entornos de nube con el fin de integrar toda la gestión y análisis del *Big Data* en una plataforma. A modo de aportar una mayor comprensión de su funcionamiento, se representa en la Figura 2.18 un esquema de los servicios de nube disponibles.

La computación en la nube se basa en el concepto de virtualización, el cual se puede definir como la tecnología que permite la creación de diferentes entornos virtuales aislados a partir de los recursos hardware disponibles y con independencia de este. Los proveedores de servicios emplean la virtualización para crear múltiples Máquinas Virtuales (VMs) y ofrecer recursos, como pueden ser el almacenamiento, la potencia de procesamiento o las aplicaciones de la nube, como diferentes servicios virtualizados. Esto aporta las siguientes ventajas a la computación en nube [43] [44]:

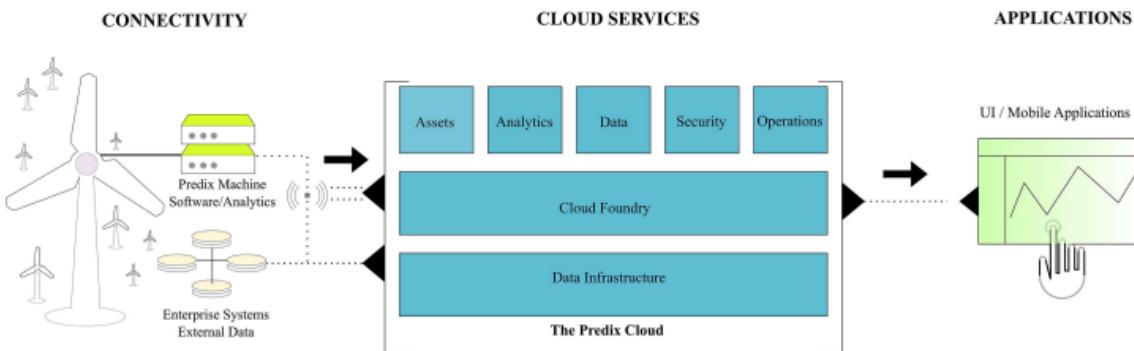


Figura 2.18: Plataforma de análisis de *Big Data* en el entorno de SGs [43]

- **Escalabilidad:** Se pueden aumentar o reducir recursos de la nube según las necesidades en cada instante temporal. Los usuarios consiguen un autoprovisionamiento de los recursos, ya que pueden desplegar instancias virtuales cuando lo deseen de forma manual o automática.
- **Flexibilidad:** Se ofrecen muchas posibilidades de configuración de los servicios de la nube.
- **Interoperabilidad:** Se emplean Interfaces de Programación de Aplicaciones (del inglés *Application Programming Interface* (API)) para integrar plataformas y aplicaciones heterogéneas. El objetivo es soportar la comunicación entre las mismas y que se produzca una interpretación de los datos de forma correcta. Por ello, también se estandarizan los protocolos para que sean comunes.
- **Seguridad:** Las plataformas de servicios de nube ofrecen un gran nivel de seguridad e implementan medidas como la encriptación de los datos y restricción de acceso o de permisos. Además, el aislamiento de los recursos y de las aplicaciones en diferentes VMs permite configurar e implementar diferentes políticas de seguridad según las necesidades.
- **Ahorro de costes:** Se implementa un modelo de pago por uso en función de los recursos que se han consumido y se evita la inversión inicial necesaria para la instalación de una infraestructura física, además de su mantenimiento en el tiempo (CAPEX y OPEX).
- **Disponibilidad:** Se garantiza el acceso a los servicios y la confiabilidad de los mismos. Para ello, implementa técnicas de redundancia y aporta una buena tolerancia a fallos.

2.5. Inteligencia Artificial

La inteligencia artificial (IA) [45] [46] se constituye como la disciplina de diseño y creación de sistemas de software o hardware de imitación de la inteligencia humana para la realización de tareas. Generalmente, estos sistemas obtienen la capacidad de aprendizaje, razonamiento, planificación y toma de decisiones a partir del entrenamiento basado en grandes cantidades de información. En otros términos, adquieren e interpretan datos del entorno o de un contexto específico y, teniendo en cuenta un objetivo determinado, procesan la información contenida para decidir la siguiente acción a realizar. Adicionalmente, son capaces de adaptar su comportamiento ante el análisis o detección de cambios en el entorno.

La IA [45] se postula como una de las tecnologías más revolucionarias y transformadoras de la actualidad y del futuro cercano por su potencial de aplicación en numerosos ámbitos. No obstante, el origen del término se remonta al año 1956, cuando el científico John McCarthy introdujo la idea de crear máquinas inteligentes tomando como base las teorías de computación de los matemáticos Norbert Wiener y John von Neumann en los años 40.

En la última década, el impulso de la IA y los grandes avances que se han producido en este campo vienen dados principalmente por el aumento de las capacidades de los equipos informáticos y el acceso a grandes cantidades de recursos computacionales en las herramientas especializadas e infraestructuras de nube. En otros términos, se facilita un almacenamiento masivo de datos y, en consecuencia, mejoras significativas en el desarrollo de algoritmos y técnicas, que cada vez se vuelven más complejas y precisas. Además, cabe destacar la contribución que han supuesto las tecnologías IoT (ver Sección 2.2) en la generación de grandes volúmenes de datos y, por tanto, en el avance de la IA al proporcionar información de valor con la que entrenar estos algoritmos.

Entrando en estas técnicas, se establecen dos divisiones dentro del campo de la IA en función de la naturaleza del aprendizaje: el automático (*Machine Learning (ML)*) y el profundo (*Deep Learning (DL)*). En las siguientes Secciones (ver Secciones 2.5.1 y 2.5.2) se expondrán las características que presentan cada una de estas ramas, además de entrar en profundidad en el funcionamiento de los modelos que se emplearán en el desarrollo de este TFM.

2.5.1. Machine Learning (ML)

La rama de aprendizaje automático (*Machine Learning (ML)*) se enfoca en el desarrollo de modelos que sean capaces de aprender y tomar decisiones en base a la introducción de datos. Durante el proceso se realizan observaciones a la información existente y, en función

de la técnica a emplear, se aplican algoritmos estadísticos para identificar patrones en los datos. Por lo tanto, mediante un entrenamiento progresivo en el tiempo, un modelo de ML puede llegar a realizar predicciones sobre datos futuros sin la intervención humana y con una alta precisión. Como se representa en la Figura 2.19, las técnicas de ML pueden pertenecer a tres categorías diferentes en función del enfoque de la información y de los objetivos que se pretenden conseguir con su aplicación [47] [48] [49]:

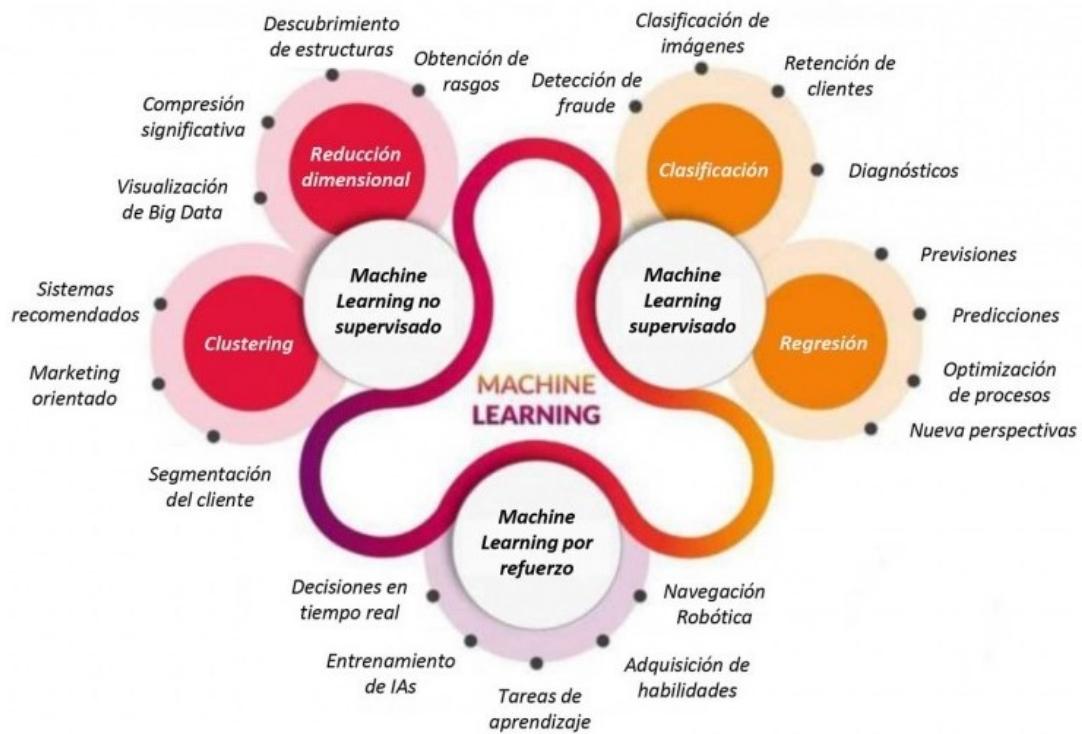


Figura 2.19: Categorías de ML [50]

- **Aprendizaje no supervisado:** Se parte de datos sin etiquetar para el entrenamiento de los modelos. Es decir, se parte del conocimiento de unos datos de entrada, pero no de la salida, por lo que el objetivo de los modelos no supervisados se basa en explorar posibles patrones en los datos mediante *clustering*.
- **Aprendizaje supervisado:** Se manejan conjuntos de datos con un conocimiento de las posibles salidas, las cuales se proporcionan en forma de etiquetas o de valores numéricos. Las técnicas de aprendizaje supervisado buscan una función óptima que consiga, dadas unas determinadas variables o características de entrada, predecir la salida con cierta precisión. Son aplicables a dos tipos de problemas: de regresión, si a partir de las variables de entrada se desea predecir un valor numérico continuo a la

salida, o de clasificación, si se predefinen diferentes clases y se asignan las variables de entrada a una de ellas.

- **Aprendizaje por refuerzo:** Las técnicas englobadas en este tipo aprenden por prueba y error, por lo que su entrenamiento a lo largo del tiempo y la experiencia permiten optimizar los resultados sin la necesidad de disponer de un gran volumen de datos.

Una vez expuestas las diferentes categorías donde se engloban las técnicas de ML, es preciso indicar que se va a centrar el estudio en el **aprendizaje supervisado** y, en particular, en la resolución de problemas de **clasificación binaria**. El motivo principal viene dado porque el objetivo que se persigue con la realización de este TFM se basa en el desarrollo de modelos que permitan detectar y predecir posibles errores que se pueden producir en una SG durante el proceso de distribución energética.

Como se detallará más adelante en la Sección 3.5.1, se creará un conjunto de datos etiquetados en función de la existencia de error o no para entrenar los modelos. Teniendo en cuenta esto, a continuación, se incluyen dos Secciones (ver Secciones 2.5.1.2 y 2.5.1.1) para presentar el funcionamiento de las dos técnicas de ML que se desarrollarán posteriormente.

2.5.1.1. Random Forest (RF)

El modelo de Bosques Aleatorios (del inglés *Random Forest* (RF)) está fundamentado en la aplicación de un conjunto de árboles de decisión. Cada uno de estos árboles se definen como estimadores y crecen mediante un proceso de entrenamiento sobre un subconjunto de datos extraído de forma aleatoria del conjunto completo. A la salida del esquema de estimadores se obtiene una predicción final basada en la votación mayoritaria de los árboles. No obstante, es un método que se puede emplear también en problemas de regresión y, en este caso, se proporcionaría un valor numérico promedio, calculado a partir de todas las salidas. En la Figura 2.20, se representa de forma gráfica el diagrama de flujo que expone el funcionamiento del modelo [51].

El proceso de construcción de los árboles de decisión se constituye por operaciones de división que van formando nodos y ramificaciones de forma iterativa. El objetivo es lograr la mayor ganancia de información posible a través de la valoración de la importancia que presentan cada una de las características del conjunto de datos. En otros términos, en cada nodo m del árbol se cuantifica la ganancia de información que presenta cada característica y, después, se selecciona la que tenga el mayor valor (j). Se aplica una división de los datos existentes en dos subconjuntos homogéneos de menor tamaño [52]:

$$\begin{aligned}\theta &= (j, t_m) \\ Q_m^{\text{left}}(\theta) &= \{(x, y) \mid x_j \leq t_m\} \\ Q_m^{\text{right}}(\theta) &= Q_m \setminus Q_m^{\text{left}}(\theta)\end{aligned}\tag{2.1}$$

Donde:

θ es la operación de división en un nodo m .

j es la característica seleccionada.

t_m es el nivel de partición de los datos.

Q_m^{left} es el subconjunto de datos en la ramificación izquierda.

Q_m^{right} es el subconjunto de datos en la ramificación derecha.

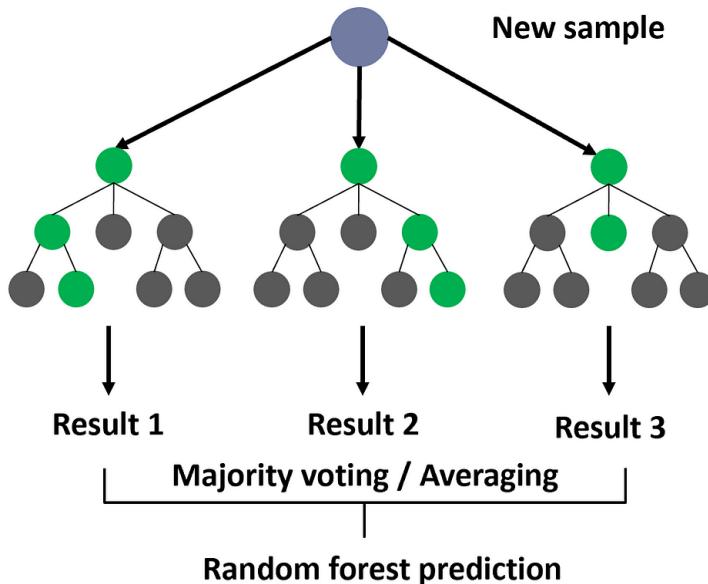


Figura 2.20: Modelo RF [51]

Este proceso se va repitiendo en todos los nodos hasta llegar al final del árbol y, en cuanto al cálculo de la ganancia de información, es preciso introducir el concepto de impureza. Este se define como el procedimiento de evaluación de la calidad de las divisiones que se producen en los nodos. En otros términos, se puede expresar que indica el nivel de ruido que hay en los datos de los subconjuntos creados mediante una función de impureza $H()$:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)) \quad (2.2)$$

Tomando esto en consideración, se pueden emplear dos funciones o métodos de medición de la impureza: [52] [53]

- Entropía: Su objetivo está orientado a establecer divisiones de los datos de forma que la entropía en los nodos inferiores o hijos sea menor que la del nodo superior o padre. Para ello, se aplica la teoría de la entropía de Shannon [52] para escoger las características que minimicen la entropía y, en consecuencia, la incertidumbre y la función de pérdidas. De forma contraria, en un nodo se obtiene una entropía máxima cuando las clases están representadas homogéneamente en el conjunto de datos.

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk}) \quad (2.3)$$

- Gini: La impureza es definida a partir del cálculo de la probabilidad de una característica determinada que es clasificada erróneamente cuando se selecciona aleatoriamente. Un índice Gini nulo expresa una clasificación pura donde todos los datos corresponden a una clase específica, en cambio, un índice Gini igual a 1, representa una distribución aleatoria de los datos.

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}) \quad (2.4)$$

Donde:

p_{mk} es la proporción de datos de entrenamiento que pertenecen a una clase determinada.

$H(Q_m)$ es la función de impureza.

2.5.1.2. Support Vector Machines (SVM)

El modelo de Máquinas de Vector Soporte (del inglés *Support Vector Machine* (SVM)) [54] [55] es una técnica que se emplea tanto en problemas de regresión como de clasificación. Se basa en el concepto del *Maximal Margin Classifier* o *Hard Margin Classifier*, puesto que tiene el fin principal de encontrar el hiperplano óptimo que consiga una máxima separación entre dos clases diferentes dentro de un espacio de características. Esta separación se define como margen y se mide como la distancia perpendicular que existe desde el hiperplano hacia los vectores de soporte, que son las muestras más cercanas a la frontera de separación

de clases. En la Figura 2.21, se representa el cálculo del mejor hiperplano entre dos clases de datos.

Por ello, se puede expresar que los vectores de soporte son los puntos más críticos del plano para asignar una de las clases predefinidas a los nuevos vectores de datos a la entrada. Este proceso de clasificación viene dado por la siguiente expresión: [56]

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq M \quad (2.5)$$

Donde:

y_i es la etiqueta de la clase.

x_i es el vector de características.

w es el vector de pesos o coeficientes del hiperplano.

b es el sesgo ajustado en el modelo.

M es el ancho definido para el margen.

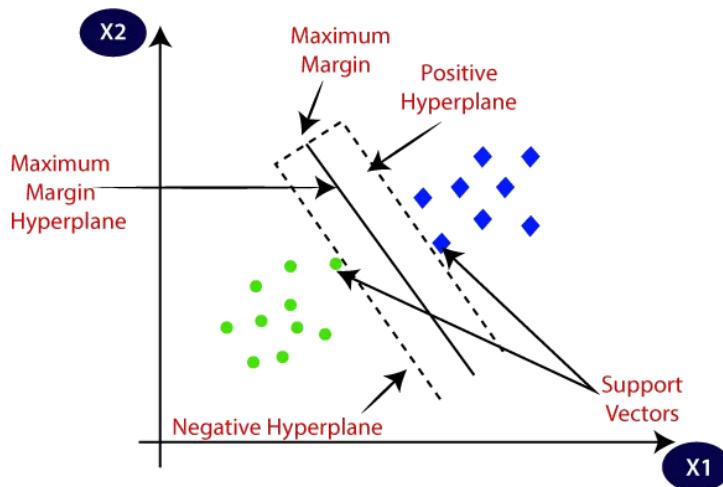


Figura 2.21: Cuantificación del hiperplano óptimo entre dos clases en el SVM [54]

No obstante, en la práctica es complicado determinar un hiperplano si no se trata de un problema ideal y basado en datos linearmente separables [57] [55]. Generalmente, en los casos reales si se intenta forzar un máximo margen, se pueden llegar a producir problemas de sobreentrenamiento, ya que nuevos datos pueden suponer grandes variaciones en el hiperplano. En la Figura 2.22, se representa la opción alternativa que se lleva a cabo en

la mayoría de casos, basada en obtener un margen máximo, permitiendo cierto grado de error (*Soft Margin Classifier*).

Esto tiene como consecuencia que existan una serie de vectores clasificados de forma errónea en el plano. En este proceso se introduce el concepto de variables de holgura (*slack variables*) a partir de la siguiente expresión:

$$\xi_i = \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) \quad (2.6)$$

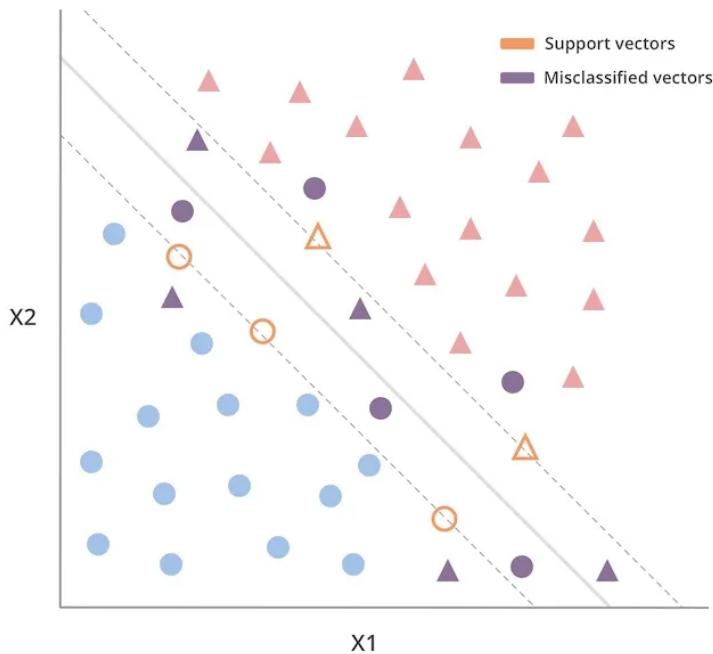


Figura 2.22: Representación de vectores clasificados erróneamente dentro de los márgenes establecidos en el SVM [56]

Estas variables determinan si un vector está correctamente clasificado o si se encuentra en una zona errónea del margen o del hiperplano, respectivamente. En el caso de posición incorrecta, el valor de la variable define la distancia al margen o el error cometido:

$$\begin{aligned} \xi_i &= 0 \text{ si } 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \leq 0 \\ 0 < \xi_i < 1 &\text{ si } 0 < 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \leq 1 \\ \xi_i > 1 &\text{ si } 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 1 \end{aligned} \quad (2.7)$$

El sumatorio de todas las variables de holgura que existen en el plano vienen incluidas en el objetivo de la función de pérdida del modelo para que el error de clasificación sea mínimo, a la vez que el margen es máximo. Como es preciso encontrar un cierto equilibrio entre ambos, el *Support Vector Machine* (SVM) se determina como una técnica de optimización convexa que se fundamenta en el hiperparámetro C . En la Figura 2.23, se representa de forma gráfica cómo su valor establece el control de forma inversamente proporcional de la cantidad de muestras clasificadas erróneamente, por lo que se define como el parámetro de regularización del modelo. Es decir, cuanto más alto sea su valor, menores violaciones del margen y del hiperplano serán permitidas y más se enfocará el modelo en un *Maximal Margin Classifier* [55].

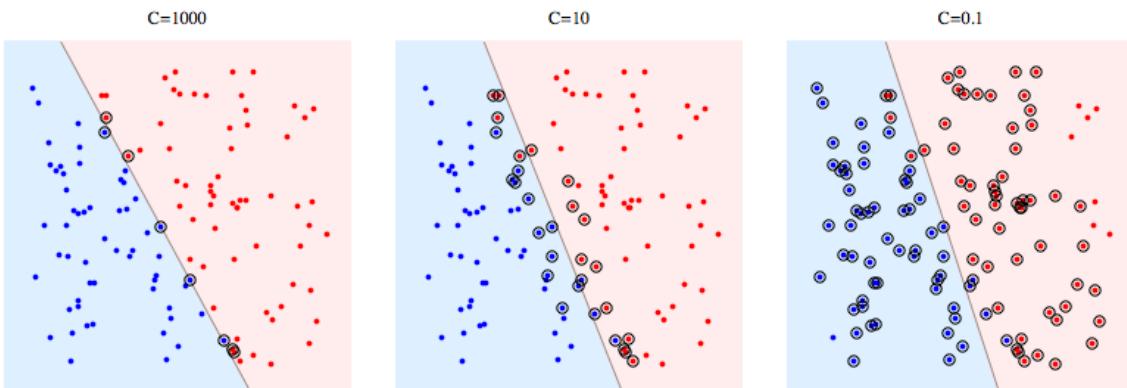


Figura 2.23: Configuración del parámetro de regularización C [58]

Además de permitir cierto grado de error, como se ha expuesto anteriormente, para enfrentar problemas con datos no linearmente separables, es imprescindible incrementar las dimensiones del espacio original de características. En este caso, se introduce el concepto de kernel como función para obtener un nuevo espacio dimensional diferente al original donde exista una mayor probabilidad de que los datos sí sean linearmente separables. La aplicación de los diferentes tipos de kernels que permiten las SVM se caracteriza a partir de las expresiones expuestas a continuación [55] [58]:

$$\begin{aligned}
 & \text{Kernel lineal: } K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \\
 & \text{Kernel polinómico: } K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d \\
 & \text{Kernel radial o gaussiano: } K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \\
 & \text{Kernel sigmoid: } K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)
 \end{aligned} \tag{2.8}$$

Para obtener una mayor compresión, se representa de forma adicional la Figura 2.24. En la misma se puede apreciar gráficamente cómo se produce la transformación dimensional de las características, especificando para el caso del kernel gaussiano o *Radial Basis Function* (RBF).

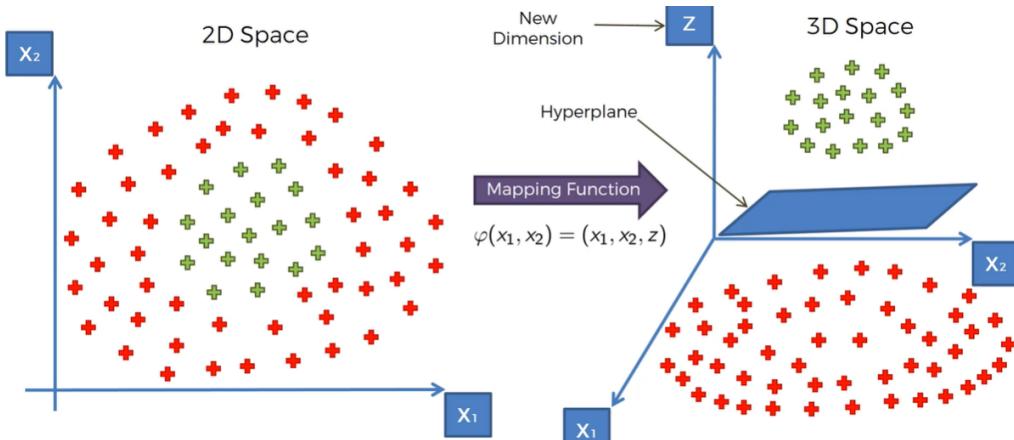


Figura 2.24: Aplicación del kernel RBF [59]

2.5.2. Deep Learning (DL)

El campo del aprendizaje profundo (*Deep Learning* (DL)) se enfoca principalmente en la creación de redes neuronales que imiten el comportamiento y la estructura lógica que tiene el cerebro humano para buscar patrones en los datos. A diferencia de las técnicas de ML, el DL se vuelve mucho más eficiente en el análisis de grandes volúmenes de datos, puesto que generalmente el ML presenta ciertas limitaciones [50] en este aspecto.

De la misma forma, el DL presenta un mejor funcionamiento en la identificación de patrones cuando se manejan características complejas o un gran número de ellas, aportando resultados con una mayor precisión que el ML. Es por ello que las técnicas de DL cobran una gran importancia en entornos donde los datos no son estructurados, como se produce en el caso de las imágenes, texto y audio. En este caso, se pueden introducir en aplicaciones de reconocimiento de voz, procesado de lenguaje natural (del inglés *Natural Language Processing* (NLP)) o visión artificial para la detección de objetos y clasificación de imágenes. [48]

De la misma forma, a nivel estructural, las diferencias entre las técnicas de ML y las de DL radican principalmente en el tratamiento de las características o variables de los datos. En el caso de emplear ML, si se manejan conjuntos con un gran número de características, sería necesario introducir un paso previo de diseño y selección de las más relevantes. Esto,

como se puede apreciar en la Figura 2.25, no ocurre en el desarrollo de un modelo de DL, ya que el tratamiento se produce internamente dentro de la red reuronal [60].

Una vez presentada la rama de aprendizaje profundo, se introduce una Sección dedicada a la exposición del modelo de DL que se desarrollará para lograr los objetivos de este TFM (ver Sección 2.5.2.1).

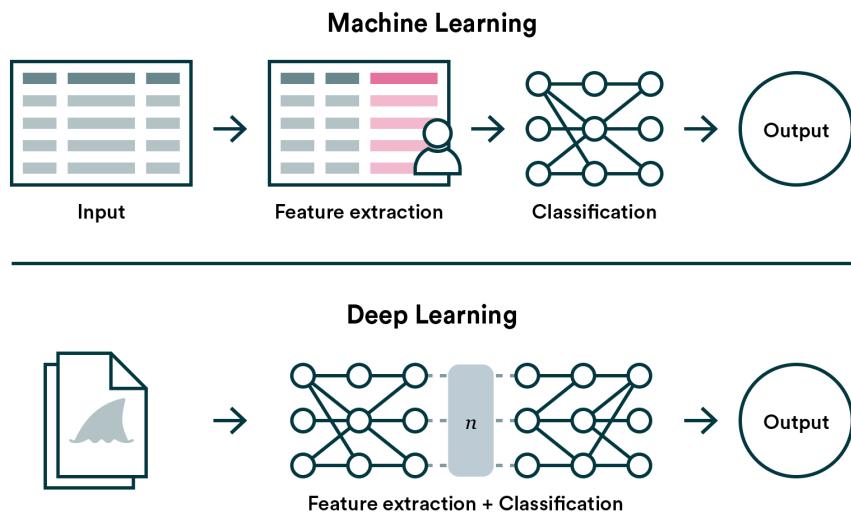


Figura 2.25: Diferencias estructurales entre el ML y el DL [60]

2.5.2.1. Red Neuronal Artificial (ANN) y Perceptrón Multicapa (MLP)

Las redes neuronales artificiales (ANN) y, en particular, los perceptrones multicapa (del inglés *MultiLayer Perceptron* (MLP)) [61], se construyen a partir de distintas capas de nodos y conexiones que replican la estructura neuronal que tiene el cerebro humano. Como se representa en la Figura 2.26, se define una capa de entrada (*input layer*), una o múltiples capas ocultas (*hidden layer*) y una capa de salida (*output layer*). El entrenamiento de la arquitectura de neuronas viene dado por las conexiones que se van estableciendo entre los nodos de las capas y los pesos y los valores de umbral que se van especificando durante el proceso.

En otros términos, se toma cada nodo como una aplicación del modelo de regresión lineal, donde se parte de unos datos en la capa de entrada, que son ponderados con unos determinados pesos en cada capa oculta de la siguiente forma:

$$z = \mathbf{w}^T \cdot \mathbf{x} + b \quad (2.9)$$

Donde:

z es la salida de cada capa oculta.

w es el vector de pesos.

b es el sesgo.

x es el vector de entrada

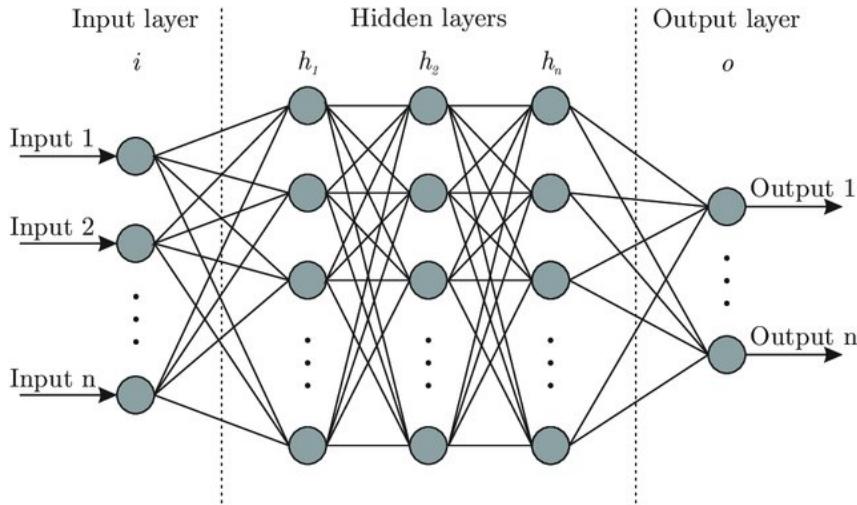


Figura 2.26: Arquitectura de un MLP [62]

De la misma forma, en cada capa oculta se aplica una función de activación o de umbral, que impone un valor límite a la salida final del nodo en cuestión y, en consecuencia, a la entrada del nodo siguiente al que se encuentre conectado. Las funciones de activación principales que se pueden emplear se representan en la Figura 2.27 y vienen dadas por las siguientes expresiones [63] [64]:

$$\begin{aligned} \text{Función Sigmoidal (Logística): } f(x) &= \frac{1}{1 + e^{-x}} \\ \text{Función ReLU (Rectified Linear Unit): } f(x) &= \max(0, x) \\ \text{Función tanh (Tangente Hiperbólica): } f(x) &= \tanh(x) \\ \text{Función Softmax: } \text{softmax}(x_i) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \end{aligned} \quad (2.10)$$

Teniendo todo el procedimiento anterior en cuenta, se puede expresar que, a partir de una serie de características, el objetivo es escoger aquellas que contribuyan a obtener una mayor precisión en la clasificación de datos de entrada futuros, puesto que se van estableciendo las conexiones óptimas entre los nodos de las capas ocultas con un entrenamiento progresivo. Es preciso indicar que el funcionamiento del modelo se basa en una red de propagación hacia delante, lo que supone que se defina un flujo unidireccional, desde la entrada a la salida.

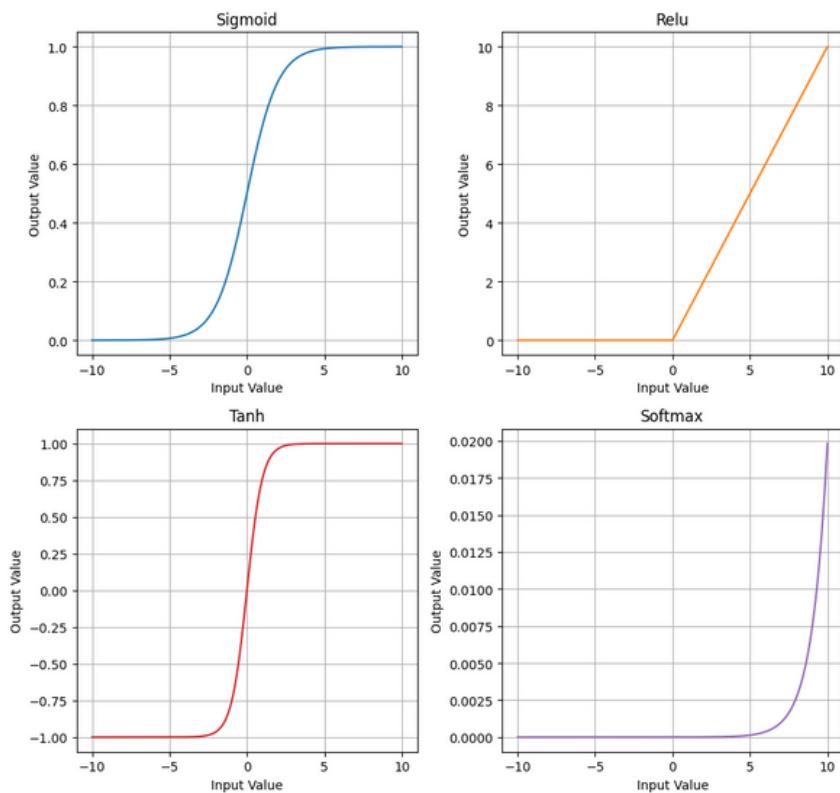


Figura 2.27: Funciones de activación [64]

No obstante, para ajustar las ponderaciones se aporta una retroalimentación, que depende directamente de la función de coste y de la salida del modelo en cada ejecución o *epoch*. Principalmente, se pueden utilizar dos métodos para minimizar la función de coste: el algoritmo adaptativo *adam* (del inglés *Adaptive Moment Estimation*), que presenta una tasa de aprendizaje adaptativa para cada parámetro, y el gradiente descendiente (del inglés *Stochastic Gradient Descent (SGD)*), que actualiza los pesos de la red en función del gradiente de la función de pérdida con respecto a los pesos [61].

2.6. Herramientas software

2.6.1. BRITE

BRITE [4] se presenta como una plataforma dedicada a la generación de topologías aleatorias de red. Fue desarrollada en la Universidad de Boston y se caracteriza por su gran flexibilidad, ya que enfoca su arquitectura en el concepto de modelo topológico. Es decir, soporta varios modelos diferentes y cada uno de ellos viene determinado por los parámetros de entrada que se definen. Es por ello que BRITE sigue la siguiente secuencia de acciones para diseñar las topologías:

1. Definición del posicionamiento de los nodos.
2. Nivel de Interconexión de los nodos y configuración de enlaces.
3. Asignación de atributos y características de la red y de los dispositivos, determinando el delay y el ancho de banda de los enlaces.
4. Especificación del formato de salida (.brite) para todas las topologías generadas.

En la Sección 3.4 se definirán las acciones realizadas con esta herramienta en el caso de este TFM. Por otro lado, con el motivo de facilitar el empleo de BRITE, se incluirá una Sección dedicada al proceso de instalación, configuración y ejecución en el Anexo correspondiente a los manuales de usuario (ver Sección C.1). Se podrá acceder a más información desde el repositorio² del equipo de investigación NetIS de la UAH.

2.6.1.1. Definición de topologías

La herramienta BRITE basa el proceso de creación de topologías en la definición de los siguientes parámetros de entrada en un fichero con formato .conf:

- *Name*: Modelo de la topología.
- *N*: Número de nodos de la topología.
- *HS y LS*: Dimensiones del plano. Respectivamente, hacen referencia a la longitud total del plano cuadrado y al tamaño de los cuadros interiores.
- *Node Placement*: Posicionamiento de los nodos. Se puede producir de forma totalmente aleatoria o creando zonas a lo largo del plano con mayor concentración de nodos.

²<https://github.com/NETSERV-UAH/BRITE>

- *Growth Type*: Método de introducción de los nodos en la topología. Se puede realizar este proceso de forma incremental (uno a uno) o de forma aleatoria (todos a la vez).
- m : Número de enlaces por nodo o número de nodos vecinos a los que se conectará un nuevo nodo al unirse a la red. BRITE puede crear enlaces unidireccionales o bidireccionales y, respectivamente, la topología generada tendría un grado m o $2m$.
- *Alpha, Beta*: Parámetros específicos para topologías basadas en el modelo Waxman.
- *BWDist, BWMin, BWMax*: Ancho de banda de los enlaces.

2.6.1.2. Modelos de topologías

Como se ha introducido, con BRITE se posibilita el uso de múltiples modelos para crear topologías. En la Figura 2.28, se representan todos los que soporta la herramienta. No obstante, este TFM se va a enfocar en la generación de topologías a nivel de router, como son los modelos Router Waxman [65] y Router Barabasi-Albert [66]. Se generan ejemplos de topologías de los mismos mediante la ejecución de un script en *python* y se representan en las Figuras 2.29 y 2.30, respectivamente.

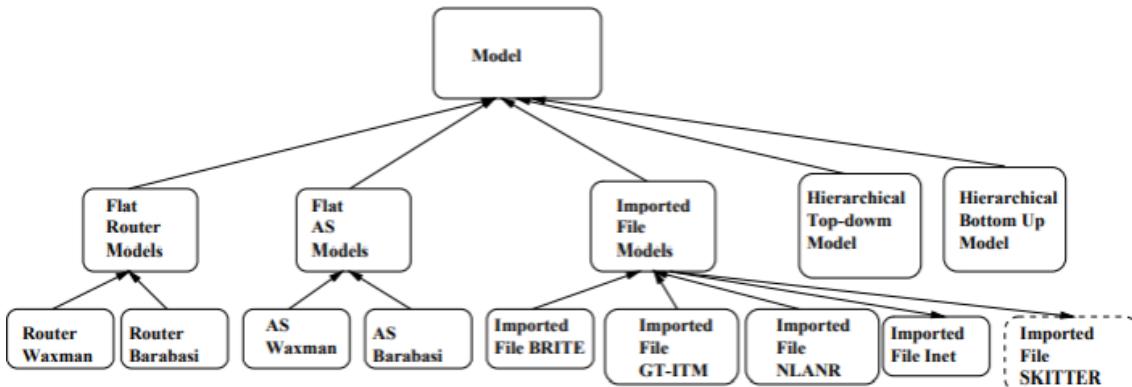


Figura 2.28: Modelos soportados por BRITE [4]

Por un lado, el modelo Router Waxman (ver Figura 2.29), como su nombre indica, emplea un modelo de probabilidad Waxman para establecer la Interconexión de los nodos en la topología [67]:

$$P_{\text{Waxman}}(u, v) = \alpha \cdot e^{-\frac{d}{\beta \cdot L}} \quad (2.11)$$

Donde:

$P(u,v)$ es la probabilidad en función de la distancia euclíadiana entre un nodo u y un nodo v de la red.

α es un parámetro específico del modelo que hace referencia a la densidad de enlaces y toma generalmente un valor igual a 0,2.

β es un parámetro específico del modelo que hace referencia al ratio enlaces largos/enlaces cortos en la topología y toma generalmente un valor igual a 0,15.

L es la máxima distancia entre dos nodos cualesquiera.

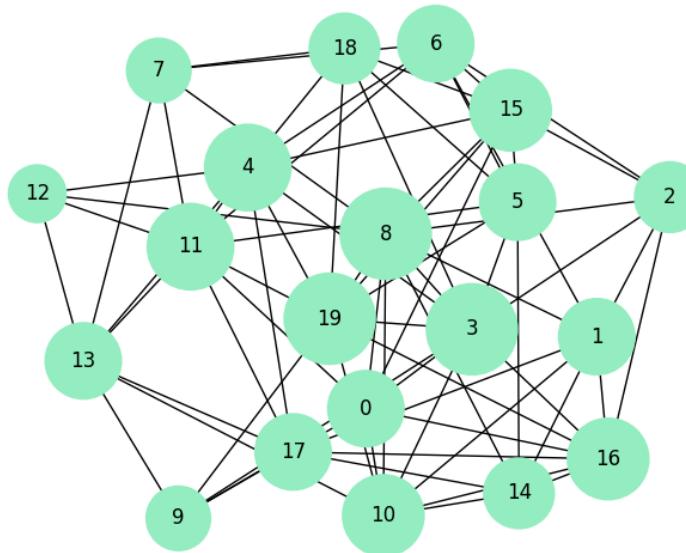


Figura 2.29: Ejemplo de topología Router Waxman

Por otro lado, el modelo Router Barabasi-Albert (ver Figura 2.30) está basado en la generación de topologías aleatorias con un incremento exponencial del número de nodos a lo largo del tiempo, también conocidas como *scale-free* o de tipo *hub-and-spoke*. Además, se permite una conexión preferencial, suponiendo que cuanto mayor grado de conectividad abarque un nodo, mayor será la probabilidad de que este añada nuevos enlaces. Por lo tanto, el plano de la topología comienza con un número de nodos inicial N_0 y se van añadiendo los demás uno a uno. Cada uno de estos nuevos nodos se conectarán a N nodos ya añadidos a la topología con una probabilidad [67]:

$$P_{\text{Barabasi-Albert}}(k_i) = \frac{k_i}{\sum_j k_j} \quad (2.12)$$

Donde:

$P(k)$ es la probabilidad de conexión del nodo i con grado k .

$\sum_j k_j$ es el sumatorio de los grados de todos los nodos de la topología.

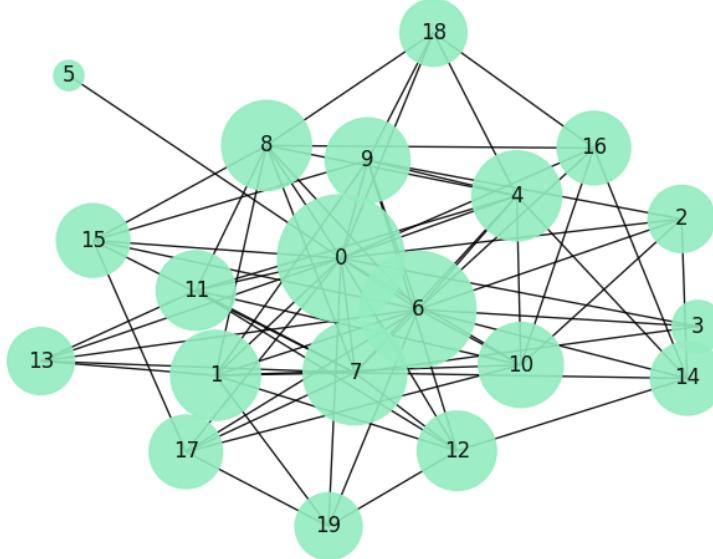


Figura 2.30: Ejemplo de topología Router Barabasi-Albert

2.6.1.3. Automatización de la ejecución

A modo de facilitar la ejecución de la herramienta BRITE se aporta en el repositorio el fichero de python *generador_brite.py*, dedicado a la automatización del proceso de creación de un fichero de configuración. Este recibirá los parámetros de entrada que caracterizarán a la topología a generar (ver Sección 2.6.1.1) y escribirá el nuevo fichero.

También, se añade el fichero de python *parser.py*, que se encargará de definir la función de transformación del archivo de salida proporcionado por BRITE (en formato .brite) en dos nuevos ficheros: *Nodos.txt* y *Enlaces.txt*. Respectivamente, estos almacenarán las posiciones x e y de los nodos en el plano y la información sobre las distancias y los identificadores de los nodos que se interconectan con cada enlace. Este funcionamiento se muestra de forma gráfica en la Figura 2.31.

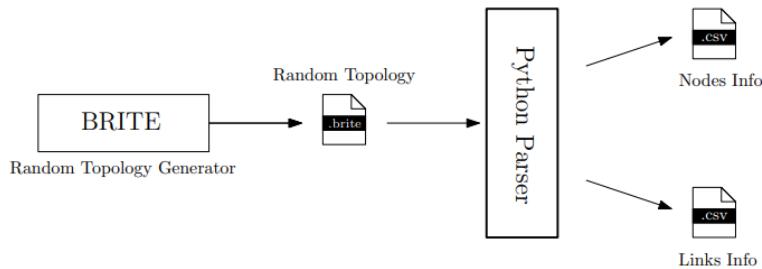


Figura 2.31: Esquema de funcionamiento de BRITE y del *parser* [3]

Teniendo en cuenta los ficheros anteriores, se proporciona un script *autogenerador.sh* donde se incluyen todas sus funcionalidades y se automatiza todo el proceso de ejecución de la herramienta. Este script sigue la siguiente secuencia de pasos:

1. Define los valores de cada uno de los parámetros de entrada.
2. Ejecuta el fichero de python *generador_brite.py* aplicando los parámetros definidos para el modelo Waxman y Barabasi.
3. Genera 10 topologías aleatorias para cada archivo de configuración a partir de 10 ficheros de semillas (*seed_files*) dados en el repositorio. Esto posibilita la generación de 10 topologías totalmente diferentes para cada escenario de red definido (a partir de unos mismos parámetros de entrada).
4. Se aplica un bucle en función del número de semilla:
 - a) Crea el directorio para cada escenario y se ejecuta la herramienta BRITE sobre cada uno.
 - b) Se ejecuta *parser.py* para obtener a la salida los ficheros *Nodos.txt* y *Enlaces.txt* de cada uno de los escenarios.

3. Diseño y análisis de datos

Este Capítulo abordará la fase del TFM que se centra en el análisis, diseño y procesado de las fuentes de datos para lograr el objetivo principal de generar un conjunto de datos final que suponga la base del entrenamiento y el desarrollo de los modelos de ML que se plantearán en el Capítulo 4. Se abrirá el Capítulo con una Sección enfocada al estudio de la disponibilidad de fuentes de datos de implementaciones reales de SGs. Por ello, será imprescindible llevar a cabo una investigación exhaustiva y un posterior análisis de su utilidad en referencia a las necesidades que presenta este TFM (ver Sección 3.1).

Por consiguiente, se añadirá una Sección en referencia al estudio de la generación energética (ver Sección 3.2). Esta vendrá justificada por la necesidad de realizar un análisis sobre algunas herramientas de simulación existentes que proporcionan datos de potencia. Este proceso posibilitará obtener información adicional sobre los recursos solares y las condiciones climáticas de una ubicación determinada. Una vez expuestas y seleccionadas las fuentes de datos que servirán como base para el desarrollo de este TFM, se añadirá una Sección dedicada a la definición detallada de las acciones que serán necesarias para llevar a cabo un procesamiento exhaustivo de toda la información recopilada (ver Sección 3.3).

Después, siguiendo las especificaciones expuestas anteriormente en la Sección 2.6.1, se plantearán una serie de escenarios mediante la generación de múltiples topologías con la herramienta BRITE (ver Sección 3.4). Los resultados que se obtengan a partir de ella se constituirán como la base sobre la que se realizarán las simulaciones en el algoritmo DEN2NE. En esta Sección (ver Sección 3.5.1) será preciso detallar las modificaciones y ajustes a implementar en el algoritmo. Esto permitirá extraer a la salida del mismo el conjunto de datos final sobre el que se entrenarán los modelos de ML y DL.

3.1. Análisis de los *datasets* disponibles

En ámbitos como el Procesado de Lenguaje Natural (del inglés *Natural Language Processing* (NLP)) o el reconocimiento facial, la disponibilidad de numerosas fuentes de datos ha sido imprescindible para acelerar el desarrollo de técnicas de minería de datos y de aprendizaje automático (ML). A diferencia de estos campos, el contexto energético experimenta

un menor grado de disponibilidad de conjuntos de datos. La protección de la privacidad de los usuarios en la medición y análisis de su comportamiento energético reduce el número de *datasets* públicos disponibles, ya que dependen principalmente de las empresas energéticas. No obstante, la búsqueda de la optimización de la distribución energética y de la reducción del consumo de los usuarios, aparte de otras motivaciones ambientales, ha instado en los últimos años a múltiples ingenieros y científicos a lo largo del mundo a crear datasets públicos enfocados a la investigación (ver Tabla 3.1).

En este contexto han cobrado gran importancia las técnicas de Monitorización de Cargas no Intrusiva (del inglés *Non-Intrusive Load Monitoring* (NILM)). Como se ha introducido en la Sección 2.2.1, cada vez se produce una mayor integración de las tecnologías de IoT en el ámbito de las SGs para obtener una mayor monitorización del comportamiento energético en los hogares. Es por ello que el fin principal de las técnicas de NILM reside en establecer una desagregación energética para poder estimar de una forma precisa el consumo individual de cada uno de los dispositivos y electrodomésticos que hay en una vivienda. Para ello, se instalan medidores en cada circuito y después, se analizan, tanto a nivel interno como a nivel externo de la vivienda, los cambios de los parámetros eléctricos en cada instante temporal [68] [69].

Teniendo esto en cuenta, se puede expresar que las técnicas de NILM se caracterizan por su bajo coste y por su facilidad y flexibilidad de despliegue. No obstante, el proceso de medición y recolección de datos puede llegar a requerir mucho tiempo y esfuerzo y, por ello, en el contexto de la investigación es importante que los datos tengan una disponibilidad pública para progresar en el desarrollo de técnicas de minería de datos y de aprendizaje automático en el ámbito energético.

En la Tabla 3.1 se exponen los múltiples datasets residenciales que han sido estudiados en esta fase de diseño, junto con información relativa a la implementación real sobre la que se ha basado cada uno de ellos, como es la ubicación de la misma, el número de viviendas total, el lapso temporal que comprende el despliegue, la frecuencia de las muestras tomadas y los parámetros eléctricos medidos. En virtud de las motivaciones y objetivos del presente TFM, es importante tener en cuenta ciertos requisitos para evaluar los conjuntos de datos disponibles y seleccionar el más apropiado:

- **Cantidad:** En primera instancia, es imprescindible revisar la cantidad de datos que aporta cada conjunto. En otros términos, para llevar a cabo un análisis estadístico del comportamiento energético de una ubicación, se requiere partir de un conjunto de datos que agrupe las mediciones de un gran número de edificios residenciales. Adicionalmente, este análisis debe comprender extensos períodos temporales de me-

dición, que permitan visualizar de forma correcta el comportamiento de los usuarios y predecir patrones de consumo a lo largo del tiempo.

- **Calidad:** Se debe evaluar la calidad de los datos recogidos, la cual responde con la resolución de las medidas que se toman. Como se puede observar en la Tabla 3.1, algunos datasets como *REDD* y *BLUED* se centran en monitorizar los edificios residenciales a una frecuencia de muestreo alta. En el contexto de las tecnologías de NILM, esto aporta una vista más representativa del comportamiento energético a nivel interno en una vivienda y permite una desagregación energética más precisa. Por lo tanto, cuanto mayor sea la frecuencia de adquisición de medidas, mayor será la resolucion de los datos.
- **Ubicación:** En términos de la ubicación de la implementación real sobre la que se han adquirido las mediciones, es de importancia conocer a la hora de analizar los datos las diferencias de voltaje que se manejan en cada país. Por ejemplo, en el caso de datasets como *BLUED* o *Smart**, los cuales provienen de ciudades de Estados Unidos, trabajarán a tensiones menores de 120V, mientras que otros como *ECO* o *GREEND*, cuyos datos se han adquirido en países europeos, lo harán con tensiones de hasta 230V [69] [70].
- **Parametrización:** Por lo general, un dataset dedicado a las tecnologías de NILM estará constituido por una colección de muestras de voltaje (V), corriente (I) y potencia activa (P), reactiva (Q) y aparente (S). Cada una de estas muestras vendrá asociada a una marca de tiempo y a un identificador que haga referencia a la vivienda o medidores a los que corresponde. Adicionalmente, algunos de los datasets que se exponen en la Tabla 3.1, como son *HUE* o *SustDataED*, también incluyen información relativa a parámetros ambientales. Esto, sobre todo en el contexto de las SGs, proporciona información fundamental sobre el impacto de las condiciones climáticas en el comportamiento de los usuarios.
- **Generación fotovoltaica:** Considerando el contexto de SGs en el que se engloba este TFM, es de vital importancia seleccionar un conjunto de datos que aporte información relativa a la generación de energía a través de fuentes renovables, como ocurre en el caso de *Smart** o *SustDataED*. En relación con esto, se expondrá en la Sección 3.2 el estudio de una plataforma específica dedicada a la adquisición de datos de generación energética.

Nombre	Localización	Nº residencias	Período de medición (días)	Resolución de medición	Parámetros
<i>AMPds2</i> [71]	Vancouver (Canadá)	1	730	60s	I, V, P, S, F, pf
<i>BLUED</i> [72]	Pittsburg (Estados Unidos)	1	8	83,33us	I, V, eventos de switch
<i>ECO</i> [73] [69]	Suiza	6	244	1s	P
<i>GREEND</i> [69]	Italia y Austria	9	310	1s	P
<i>HUE</i> [74]	British Columbia, Canadá	28	60	1s	P
<i>iAWE</i> [75]	Nueva Delhi (La India)	1	73	1s	V, I, P, S
<i>REDD</i> [76]	Boston (Estados Unidos)	6	119	66,66us	I, V, P
<i>Smart*</i> [77]	Massachussets (Estados Unidos)	3	90	60s	P, S, V, I
<i>SustDataED</i> [78]	Madeira (Portugal)	50	1144	60s	I, V, P, Q, S
<i>UK-DALE</i> [79]	Reino Unido	5	499	62,5us	P, estado de switch

Tabla 3.1: Comparación entre datasets públicos en el ámbito NILM [69] [80] [81] [70]

Como se ha introducido en el Capítulo 1, el objetivo principal que se pretende con este TFM es desarrollar técnicas de ML y DL para identificar y predecir de forma precisa los errores que se pueden producir en una SG durante el proceso de distribución energética. A partir de ello, se expresa la necesidad de seleccionar un conjunto de datos en el que se pueda analizar el comportamiento de múltiples usuarios, tanto de consumo como de producción, durante un extenso período de tiempo.

De igual manera, es importante que las muestras tengan una buena resolución y que proporcione información en cada instante sobre las condiciones climáticas de la ubicación donde se encuentran las viviendas. Bajo las premisas anteriores, se ha finalizado el proceso de estudio y análisis de los diferentes conjuntos de datos llegando a la conclusión de que el dataset más adecuado a emplear en este TFM es ***SustDataED***.

3.1.1. ***SustDataED***

El dataset *SustDataED* [78] es creado a partir del proyecto de investigación *Sustainable Interactions with Social Networks, context Awareness and Innovative Services* (SINAIS), dedicado al diseño, implementación y despliegue de sistemas NILM de bajo coste. Surge con el objetivo de proporcionar una retroalimentación ecológica a los usuarios para fomentar un comportamiento energético sostenible y un mayor uso de las fuentes de energía renovables. Como conjunto de datos, engloba cinco años de datos de consumo y producción energética de 50 hogares de la ciudad de Funchal (Madeira, Portugal) con una resolución o frecuencia de adquisición de nuevas muestras por minuto. Debido a esto, se caracteriza por comprender una gran cantidad de información eléctrica que tendrá que ser posteriormente procesada de forma exhaustiva (ver Sección 3.3).

Pese a la necesidad de procesamiento que conlleva, el uso de *SustDataED* en el presente TFM brinda ciertas ventajas. Entre otras, aporta un gran perspectiva del comportamiento eléctrico a largo plazo de múltiples hogares con diferentes características y ofrece un enfoque medioambiental, ya que mediante la adquisición de los datos relativos a las condiciones climáticas posibilita el análisis de la correlación de las mismas con el consumo de los usuarios y la producción de energía.

3.1.2. Estructura del dataset ***SustDataED***

El proceso de adquisición y recolección de los datos de los 50 hogares que constituyen el dataset se divide en cuatro despliegues¹ diferentes (ver Figura 3.1):

¹Período temporal de implementación y operación de un sistema de adquisición de datos en un grupo de hogares para recopilar información energética

- Los despliegues 1 y 2 se pueden tratar en conjunto, ya que abarcan las mismas viviendas, suponiendo un total de 23. El despliegue 1 se inicia en julio de 2010 y finaliza en el mes de noviembre de ese mismo año cuando se reinstala una versión revisada del sistema de realimentación. En este momento, comienza el despliegue 2, que durará hasta abril del 2012.
- El despliegue 3 se implementa en un total de 17 hogares desde el mes de agosto de 2012 hasta enero de 2013. Se introducen novedades en el sistema de adquisición de datos para reducir el número de sensores a instalar, optimizando la arquitectura de monitorización.
- El despliegue 4 es el último que se realiza y consta de la agrupación de medidas de 10 hogares, comenzando la recolección de datos de los mismos en el mes de julio de 2013 y finalizando en marzo de 2014.

En la Figura 3.1 se representan gráficamente los períodos temporales que abarcan los despliegues expuestos anteriormente mediante el empleo de la librería de *Python*, *matplotlib*. Además, como se puede apreciar, cada uno de los 50 hogares participantes es representado por un identificador único (*id*).

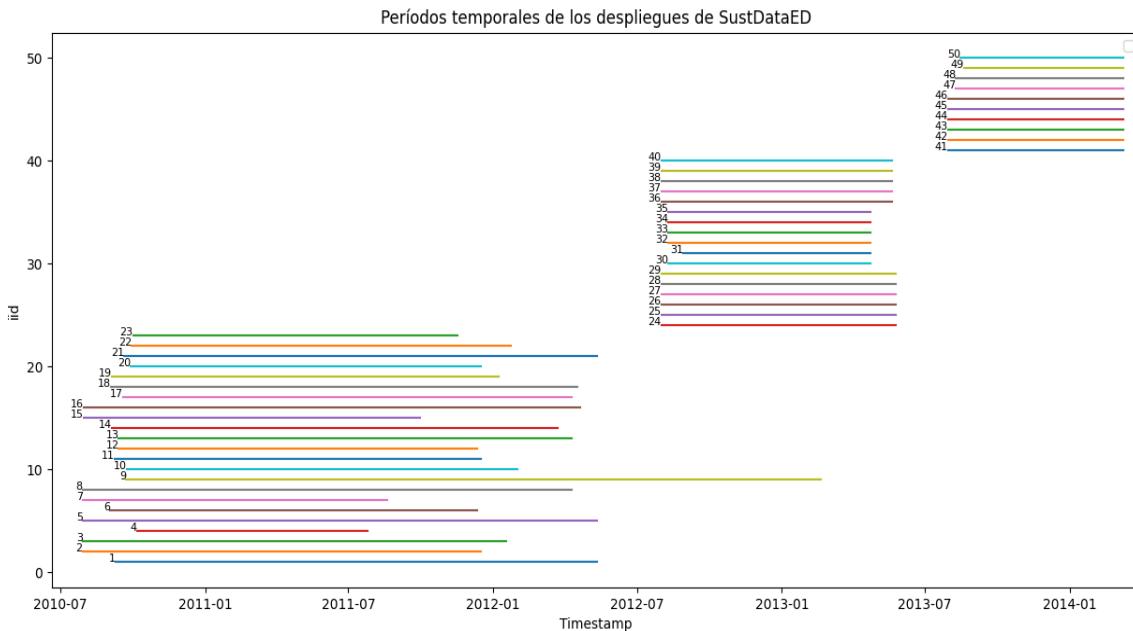


Figura 3.1: Representación del período temporal que comprende el proceso de recolección de datos para cada hogar

A continuación, se detallará a modo organizativo, cómo el dataset *SustDataED* clasifica en varias tablas los datos en función del tipo de información al que se hace referencia.

3.1.2.1. Medidas de consumo de energía

En la Tabla 3.2 se listan los campos que han sido definidos para la recolección de medidas de consumo de cada uno de los hogares. El dataset asocia los parámetros eléctricos a dos identificadores: uno para describir la vivienda de la que se están recogiendo los datos, y otro, para el despliegue al que pertenece.

Por otro lado, se almacena para cada medida una marca de tiempo con la fecha, hora, minuto y segundo en la que se ha recogido la información y se incluye el parámetro *miss_flag* para indicar si para un determinado instante temporal se han adquirido datos de forma incorrecta, suponiendo filas vacías en el dataset.

Campo	Descripción	Unidades
<i>home_id</i>	Identificador de vivienda monitorizada	-
<i>timestamp</i>	Instante temporal de medida	datetime
<i>deploy</i>	Identificador del despliegue	-
<i>Imin</i>	Corriente mínima	A
<i>Imax</i>	Corriente máxima	A
<i>Iavg</i>	Corriente media	A
<i>Vmin</i>	Tensión mínima	V
<i>Vmax</i>	Tensión máxima	V
<i>Vavg</i>	Tensión media	V
<i>Pmin</i>	Potencia activa mínima	W
<i>Pmax</i>	Potencia activa máxima	W
<i>Pavg</i>	Potencia activa media	W
<i>Qmin</i>	Potencia reactiva mínima	VAR
<i>Qmax</i>	Potencia reactiva máxima	VAR
<i>Qavg</i>	Potencia reactiva media	VAR
<i>Smin</i>	Potencia aparente mínima	VA
<i>Smax</i>	Potencia aparente máxima	VA
<i>Savg</i>	Potencia aparente media	VA
<i>PFmin</i>	Factor de potencia mínimo	-
<i>PFmax</i>	Factor de potencia máximo	-
<i>PFavg</i>	Factor de potencia medio	-
<i>miss_flag</i>	Flag de valores vacíos	-

Tabla 3.2: Tabla de medidas de consumo energético [78]

Los parámetros de tensión (V) y corriente (I) se muestrean continuamente a una frecuencia de 8KHz y las medidas de potencia activa (P(W)), reactiva (Q(VAR)) y aparente (S(VA)) se cuantifican a un ratio de 50 muestras por segundo a partir de las siguientes expresiones [78]:

$$\begin{aligned} S &= I_{\text{RMS}} \cdot V_{\text{RMS}} \\ P &= S \cdot \cos(\phi) \\ Q &= S \cdot \sin(\phi) \end{aligned} \tag{3.1}$$

Donde:

I_{RMS} es la corriente eficaz (RMS).

V_{RMS} es el voltaje eficaz (RMS).

ϕ es el ángulo de fase entre la corriente y el voltaje instantáneos.

No obstante, como se había introducido en el Apartado 3.1.1 y se había definido en la Tabla 3.1, todas estas medidas, una vez son recogidas, se almacenan en el dataset a una resolución de una muestra por minuto. Se introduce también en la Tabla 3.2 el valor de factor de potencia (PF) para medir la eficiencia del uso de energía en un sistema eléctrico. Se cuantifica a partir de la razón entre la potencia activa (P) y la potencia aparente (S), siendo ideal un valor de 1.

3.1.2.2. Medidas de eventos de potencia

La información sobre los eventos de potencia se extrae de las medidas en crudo de potencia activa que, como se ha expresado en el apartado anterior, se producen a una frecuencia de muestreo de 50Hz. La Tabla 3.3 se dedica al proceso de detección de dichos eventos o, en otros términos, de identificación de cambios en la carga total de una vivienda. Como se ha introducido en la Sección 3.1, estos datos serían de gran utilidad en el caso de requerirse el empleo de técnicas de NILM.

El proceso de recolección de las muestras tiene como base un método estadístico que calcula la probabilidad de que se produzca un cambio del valor de potencia media entre dos instantes. Para ello, el algoritmo trabaja con dos ventanas temporales: una de detección, que cuantifica la probabilidad del cambio en una muestra determinada y otra de búsqueda de los valores mínimos y máximos de potencia, que define las posiciones de los eventos en la señal de potencia agregada. En referencia a esto, viene definido un cambio mínimo de potencia real de 30 W.

Campo	Descripción	Unidades
<i>home_id</i>	Identificador de vivienda monitorizada	-
<i>timestamp</i>	Instante temporal de medida	datetime
<i>deploy</i>	Identificador del despliegue	-
<i>delta_P</i>	Cambio real de potencia activa	W
<i>delta_Q</i>	Cambio real de potencia reactiva	VA
<i>trace_P</i>	Traza real de potencia activa del evento	W
<i>trace_Q</i>	Traza real de potencia reactiva del evento	VAR

Tabla 3.3: Tabla de medidas de eventos de potencia [78]

3.1.2.3. Medidas demográficas

Para el análisis del comportamiento eléctrico de cada uno de los hogares monitorizados se puede visualizar en la Tabla 3.4 cómo se almacena la información respectiva a sus inquilinos y a los períodos temporales de adquisición de datos de cada vivienda.

Campo	Descripción	Unidades
<i>home_id</i>	Identificador de vivienda monitorizada	-
<i>building_id</i>	Identificador de edificio	-
<i>begin_monitoring</i>	Fecha de inicio de medición	datetime
<i>end_monitoring</i>	Fecha de fin de medición	datetime
<i>begin_feedback</i>	Fecha de inicio de realimentación en el despliegue	datetime
<i>end_feedback</i>	Fecha de fin de realimentación en el despliegue	datetime
<i>type</i>	Tipo de vivienda	-
<i>bedrooms</i>	Número de habitaciones	-
<i>adults</i>	Número de adultos inquilinos	-
<i>children</i>	Número de niños inquilinos	-
<i>contracted_power</i>	Potencia contratada	kWh

Tabla 3.4: Tabla de medidas de producción de energía [78]

De forma adicional, el dataset *SustDataED* incluye otra tabla con datos complementarios sobre las personas participantes en los despliegues, como son el género, edad, educación y ocupación.

3.1.2.4. Medidas de producción de energía

El dataset *SustDataED* recoge a nivel global los datos respectivos a la producción de electricidad en intervalos de 15 minutos. La energía generada se desagrega según su fuente de procedencia, dando protagonismo a las fuentes renovables. En la Tabla 3.5 se indican los campos que han sido definidos para cuantificar las medidas. Adicionalmente, en la

Figura 3.2, se representan gráficamente los valores recogidos en el dataset a lo largo del tiempo para cada una de las fuentes de energía dadas. En este caso se emplea la librería de *Python*, *matplotlib*.

Campo	Descripción	Unidades
<i>timestamp</i>	Instante temporal de medida	datetime
<i>total</i>	Producción total	MWh
<i>thermal</i>	Electricidad producida por fuentes térmicas	MWh
<i>hydro</i>	Electricidad producida por fuentes hidroeléctricas	MWh
<i>eolic</i>	Electricidad producida por fuentes eólicas	MWh
<i>biomass</i>	Electricidad producida por fuentes orgánicas	MWh
<i>solar</i>	Electricidad producida por fuentes fotovoltaicas	MWh

Tabla 3.5: Tabla de medidas de producción de energía [78]

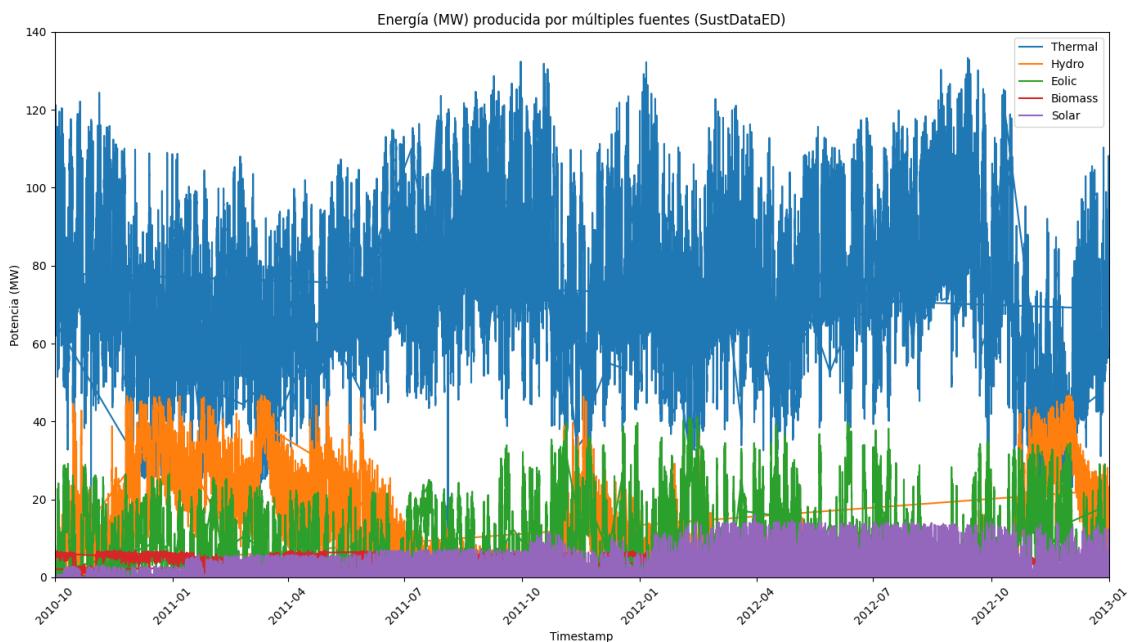


Figura 3.2: Gráfica de valores de múltiples fuentes energéticas (*SustDataED*)

3.1.2.5. Medidas de eventos de usuario

Una de las ventajas que proporciona la monitorización del comportamiento eléctrico de un hogar es la realimentación de la información que se produce hacia los mismos usuarios. El dataset recoge en forma de eventos las interacciones que se llevan a cabo con este sistema de realimentación a través de los campos determinados en la Tabla 3.6.

Campo	Descripción	Unidades
<i>home_id</i>	Identificador de vivienda monitorizada	-
<i>timestamp</i>	Instante temporal de medida	datetime
<i>deploy</i>	Identificador del despliegue	-
<i>type</i>	Tipo de interacción	-
<i>view_id</i>	Identificador de pantalla de visualización	-
<i>view_name</i>	Nombre de pantalla de visualización	-

Tabla 3.6: Tabla de medidas de eventos de usuario [78]

3.1.2.6. Medidas de condiciones ambientales y climáticas

Como se ha introducido al comienzo de la Sección 3.1.1, el dataset aporta un enfoque medioambiental a los procesos de consumo y producción energética al incluir la recolección de los datos relativos a las condiciones climáticas que se dan en cada instante. En la Tabla 3.7 se muestran los parámetros que se tienen en cuenta en las mediciones. Se incluye un campo dedicado a los eventos climáticos, que aporta información adicional cuando se produce un suceso metereológico determinado.

Campo	Descripción	Unidades
<i>timestamp</i>	Instante temporal de medida	datetime
<i>temperature</i>	Temperatura exterior	°C
<i>humidity</i>	Humedad relativa	%
<i>pressure</i>	Presión relativa	hPa
<i>wind_dir</i>	Dirección del viento	-
<i>wind_speed</i>	Velocidad del viento	kmh
<i>precipitation</i>	Niveles de precipitación	mm
<i>events</i>	Eventos relevantes	-
<i>conditions</i>	Condiciones del cielo	-

Tabla 3.7: Tabla de medidas de condiciones ambientales [78]

3.1.3. Conclusiones del análisis del dataset *SustDataED*

Una vez realizado el estudio y el análisis en profundidad de los datos proporcionados por el dataset *SustDataED*, se definen una serie de conclusiones a tener en cuenta antes de iniciar la fase de preprocesamiento de los datos. Por ello, considerando las motivaciones de la realización de este TFM, se puede expresar que la utilidad que proporciona el dataset *SustDataED* viene dada principalmente por el uso de las medidas de consumo, de producción y de las condiciones ambientales (ver Tablas 3.2, 3.5 y 3.7).

El procesamiento que será requerido para estos datos se detallará en la Sección 3.3. No obstante, es importante incidir en las características de los datos de producción eléctrica dados por *SustDataED*. Como se ha expuesto anteriormente en la Sección 3.1.2.4, el dataset proporciona esta información en términos globales y después, desagrega los valores energéticos según su fuente de procedencia. Por lo tanto, en la Sección 3.3.2 de este documento se detallará cómo será imprescindible determinar si estos datos son precisos y evaluar su utilidad específicamente en un entorno de SGs como motivo de los objetivos de este TFM.

Por otro lado, en cuanto a los eventos de potencia (ver Tabla 3.3), como este TFM no se basa en el diseño de técnicas NILM, ni se pretende establecer una desagregación de potencia a nivel interno de una vivienda, se expone este tipo de datos únicamente a modo de estudio. De la misma forma, sucede en el caso de los eventos de usuario (ver Tabla 3.6), los cuales no proporcionan información de utilidad que sea necesaria de procesar en la siguiente fase.

3.2. Simulación de datos de producción

Como se ha expuesto en el Apartado anterior de conclusiones del dataset (ver Sección 3.1.3), en la siguiente etapa de este TFM, dedicada al preprocesamiento de los datos (ver Sección 3.3), se requerirá evaluar la precisión y la utilidad de los datos de producción proporcionados por *SustDataED*. Para ello, la presente Sección viene definida por la necesidad de obtener una fuente de datos de producción energética adicional, que permita contrastar la información adquirida del dataset anteriormente.

Atendiendo a este fin, se procederá a emplear varias herramientas de análisis y simulación de datos de producción energética, detallándose a su vez, sus características de funcionamiento y la configuración necesaria. Cabe destacar que el proceso de simulación de los datos que se va a llevar a cabo implica recrear un dataset que sea riguroso con la realidad.

Por lo tanto, en primera instancia, se deberá llevar a cabo un estudio y un análisis específico de las características geográficas y climáticas a largo plazo de la localización que se tomará como base. Por consiguiente, se cuantificarán los datos mediante una herramienta de simulación en función del estudio anterior. Finalmente, se analizarán los resultados y se evaluará su precisión, con el fin de procesarlos y combinarlos con los datos adquiridos de *SustDataED* (ver Sección 3.3.2). Es decir, será en esta fase de preprocesamiento donde se tome la decisión final de selección de la fuente de datos de producción más adecuada para el desarrollo de este TFM.

3.2.1. Estudio y análisis de la ubicación (*Global Solar Atlas*)

En cuanto a la ubicación, en este caso, será preciso basarse concretamente en Funchal, capital de la isla de Madeira (Portugal), ya que el análisis a realizar tiene que ser acorde a los datos recogidos en el dataset *SustDataED*.

Según los parámetros de clasificación climática de Köppen [82], la ciudad de Funchal se caracteriza por tener un clima mediterráneo. Al localizarse en una zona subtropical, presenta oscilaciones diarias mínimas, lo que se traduce en escasos cambios de temperatura entre las diferentes estaciones del año. Por ello, los inviernos son suaves y con precipitaciones moderadas, mientras que los veranos tienden a ser ligeramente más cálidos y secos.

Tomando en consideración lo anterior, para conocer el potencial de los recursos solares que presenta la ubicación se va a proceder al empleo del modelo solar *Solargis*, proporcionado por la plataforma online *Global Solar Atlas*² [83] [84]. Esta plataforma es financiada por el *Energy Sector Management Assistance Program* (ESMAP) y administrada por la organización de El Banco Mundial con el objetivo de mapear los recursos de energía renovable a nivel global, permitiendo el acceso a una gran cantidad mapas y de datos promediados a largo plazo y en tiempo real de cualquier punto de la Tierra.

La información sobre los recursos solares y la cuantificación de la energía se suministran a través de esta plataforma siguiendo el estándar GIS ráster o cuadriculado con formatos GeoTIFF o AAIGRID. Para las diferentes capas de datos que se pueden determinar en función de los parámetros de radiación o del potencial fotovoltaico, se sigue una referencia espacial geográfica en base al código EPSG 4326 [85]. Este código es asignado por la organización *European Petroleum Survey Group* (EPSG) para identificar la proyección geográfica empleada, la cual en este caso hace referencia al sistema de coordenadas convencional (latitud-longitud) que se utiliza para la representación cartográfica de la Tierra. Por otro lado, los metadatos correspondientes a las características de cada capa se proveen en formato PDF o XML, siguiendo la estructura de datos geográficos definida por ISO 19115:2003/19139 [83] [86].

3.2.1.1. Identificación de capas de datos

Considerando las motivaciones que se persiguen con el empleo de la plataforma *Global Solar Atlas*, es preciso realizar un paso previo al análisis, basado en la identificación de los tipos de capas de datos que se pueden configurar [86]:

²<https://globalsolaratlas.info/map>

- *Direct Normal Irradiation* (DNI) (kWh/m^2): El índice de radiación directa normal se define como la cantidad de radiación solar que llega perpendicularmente a la superficie de la placa fotovoltaica, sin tener en cuenta los posibles efectos atmosféricos de dispersión o absorción.
- *Diffuse Irradiance Fraction* (DIF) (kWh/m^2): El índice de radiación difusa hace referencia a la porción de radiación dispersada por las nubes y los gases atmosféricos, lo que produce que provenga de todas las direcciones.
- *Global Horizontal Irradiation* (GHI) (kWh/m^2): El índice de radiación horizontal global viene dado por el sumatorio de la radiación solar directa (DNI) y la radiación solar difusa dispersada en consecuencia a los efectos de la atmósfera (DIF).
- *Global Irradiation at Optimum Tilt* (GTI) (kWh/m^2): El índice de radiación global inclinada se refiere al total de radiación que incide en una superficie inclinada, que generalmente se encuentra ajustada un ángulo óptimo para maximizar la captación en términos anuales.
- *Photovoltaic Power Potential* (PVOUT) (kWh/kWp): El parámetro que mide el potencial energético de los sistemas fotovoltaicos de una ubicación determinada se cuantifica a partir de un sistema de referencia construido por módulos de silicio cristalino, de 1kWp (kilovatio pico) y que se encuentra inclinado un ángulo óptimo.

Es necesario indicar que todos los parámetros anteriores se proporcionan para cada ubicación como valores promedios anuales de los totales diarios. Adicionalmente, otras capas a tener en cuenta para el análisis de los datos podrían ser la temperatura del aire, que determina en gran medida el ambiente de operación de las placas fotovoltaicas, o la elevación del terreno, que puede convertirse en un factor limitante para la instalación de plantas solares.

En las Figuras 3.3 y 3.4, se visualizan en formato GeoTIFF los resultados de aplicar las capas de los índices de radiación DNI y GHI al mapa mundial a través de la plataforma *Global Solar Atlas*. Por otro lado, en la Figura 3.5, se representa el potencial fotovoltaico dado por el parámetro PVOUT. Como es de esperar, se puede verificar que existe una gran correlación entre los niveles de radiación recibidos con respecto a la cantidad de energía que se puede generar con una planta solar en una ubicación determinada.

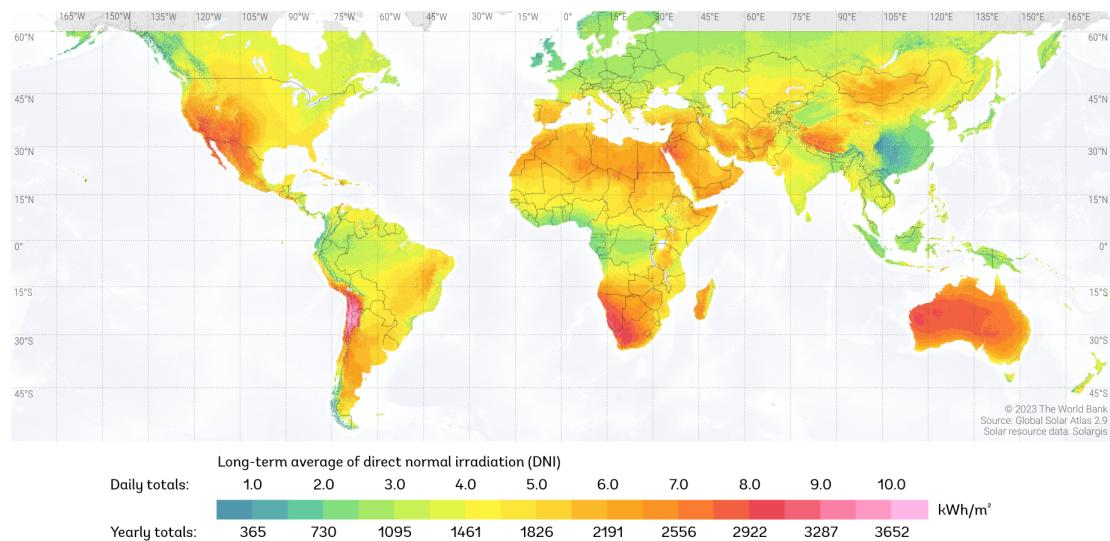


Figura 3.3: Mapa mundial del índice de radiación directa normal (DNI) mundial [83]

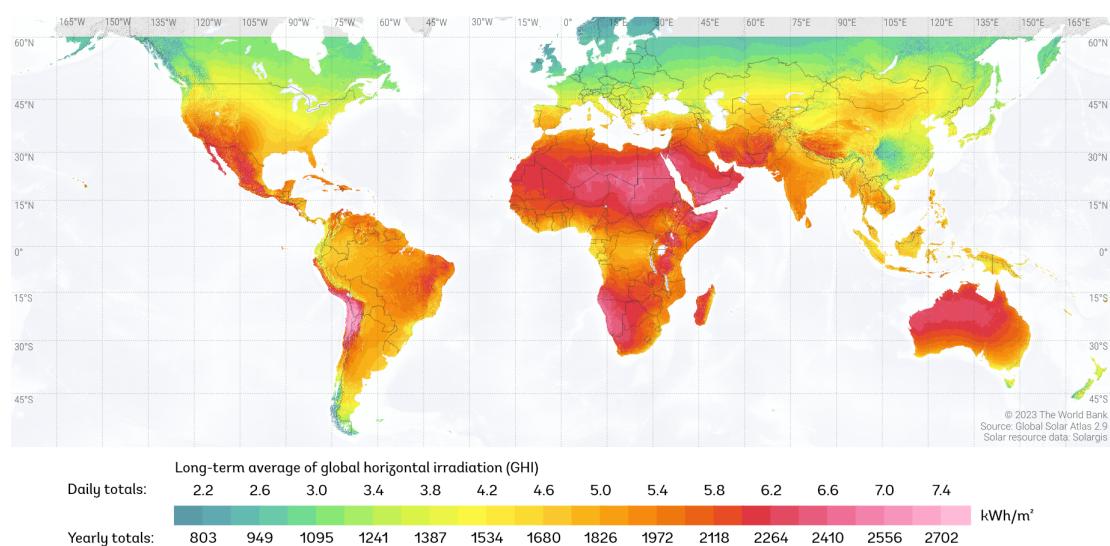


Figura 3.4: Mapa del índice de radiación horizontal global (GHI) mundial [83]

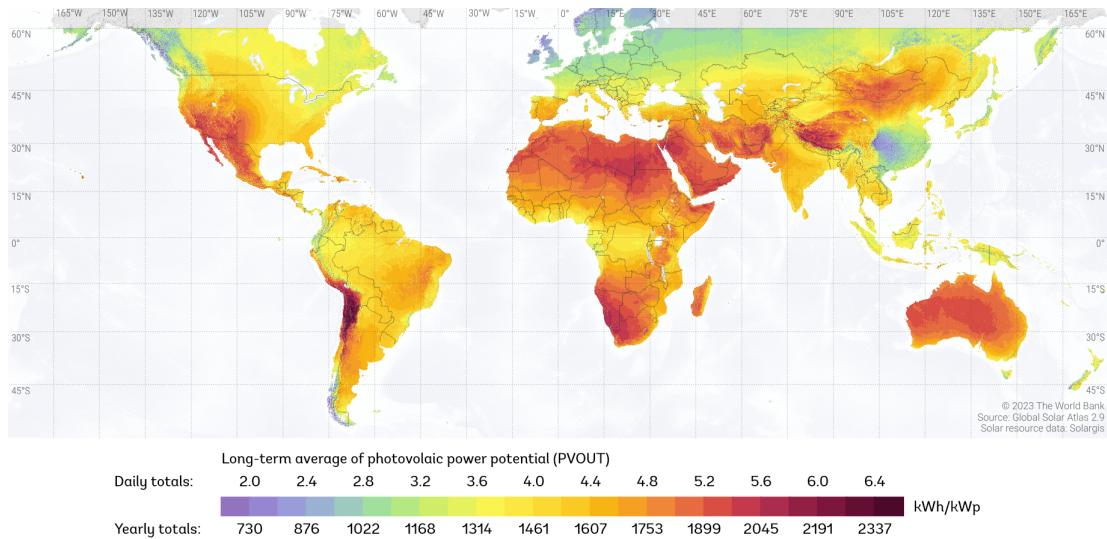


Figura 3.5: Mapa del potencial de producción energética fotovoltaica (PVOUT) mundial [83]

3.2.1.2. Análisis de resultados de radiación

De la misma forma, lo anterior se hace patente para el caso de la isla de Madeira, como se puede visualizar en las Figuras 3.6, 3.7 y 3.8. Desde la plataforma *Global Solar Atlas* también se extraen y se recogen en la Tabla 3.8 los valores de los índices de radiación, configurados específicamente para la localización de la ciudad de Funchal.

Parámetro	kWh/m ² /día	kWh/m ² /año
DNI	3,698	1349,9
DIF	2,038	743,8
GHI	4,345	1586,1
GTI	4,730	1726,3

Tabla 3.8: Tabla de valores extraídos para cada índice de radiación en la ciudad de Funchal [83]

Por tanto, se puede comprobar a través de estos valores y de la leyenda proporcionada en las figuras anteriores, que Funchal se percibe como una ciudad con un potencial de recursos solares medio alto. Esto es coherente con su localización subtropical y con la elevación del terreno. Es decir, al encontrarse en una zona de costa, ronda en valores cercanos a los 100 metros (concretamente 137 metros en la ubicación seleccionada) y se reduce ligeramente el nivel de energía solar capturable respecto a otras zonas de la isla más montañosas.

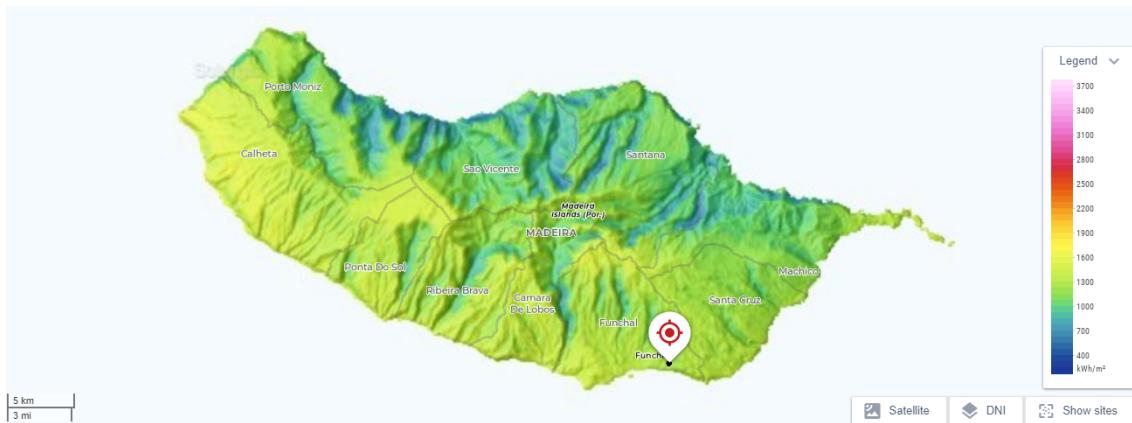


Figura 3.6: Mapa del índice de radiación directa normal (DNI) de Madeira [83]



Figura 3.7: Mapa del índice de radiación horizontal global (GHI) de Madeira [83]



Figura 3.8: Mapa del potencial de producción energética fotovoltaica (PVOUT) de Madeira [83]

3.2.1.3. Configuración y cuantificación del potencial energético de la ubicación

Para calcular de forma aproximada la cantidad de energía que supondría la instalación de un sistema fotovoltaico en la ciudad de Funchal, es preciso configurar primero los siguientes parámetros a través de la plataforma:

- Tipo y tamaño de sistema fotovoltaico: Se permite seleccionar entre un sistema enfocado a una vivienda, a un edificio comercial o a una planta solar de grandes dimensiones. En este caso, se toma como base el estudio de producción energética en un entorno residencial. Es preciso indicar que en la plataforma la configuración de un sistema fotovoltaico para un entorno residencial no considera la opción de almacenamiento de electricidad.
- Capacidad de instalación: Se toma una capacidad máxima de generación de 4kWp (kilovatios pico) para el sistema. En otros términos, este valor determina la cantidad de energía que producirían los paneles en condiciones óptimas.
- Azimut [87]: Se determina como el ángulo de orientación horizontal y, en función de su valor, define la proyección de los paneles solares en dirección norte, sur, este u oeste. Se indica un valor de 180º para establecer una orientación hacia el sur.
- Inclinación: La plataforma establece como ángulo óptimo para la localización de Funchal un ángulo de 26º, por lo que se configura la instalación de los paneles solares sobre rieles sujetos a un tejado con esta misma inclinación.

3.2.1.4. Análisis de resultados de potencia

Una vez configurados los parámetros expuestos, se obtienen como resultados la trayectoria solar diaria y los perfiles de radiación y generación fotovoltaica diarios y mensuales para la ubicación de Funchal. En la Figura 3.9 se representa la trayectoria solar diaria, en la que se puede visualizar la comparación de la elevación que percibe el sol en función del instante del año, siendo máxima durante el solsticio de junio, y mínima, durante el de diciembre. En el caso del equinoccio, se sigue una curva con valores promedios, al existir una igualdad entre las horas de día y las de noche.

Por otro lado, en la Figura 3.10, se modelan gráficamente los valores totales mensuales, tanto de producción energética, como de radiación directa normal DNI para visualizar la correlación existente entre los mismos en los distintos meses del año. El valor promedio máximo energético se alcanza en el mes de julio, con un valor de 565kWh, ocurriendo de la misma forma para el valor máximo de radiación, cuyo alcance es de 163,7kWh/m² para este mes.

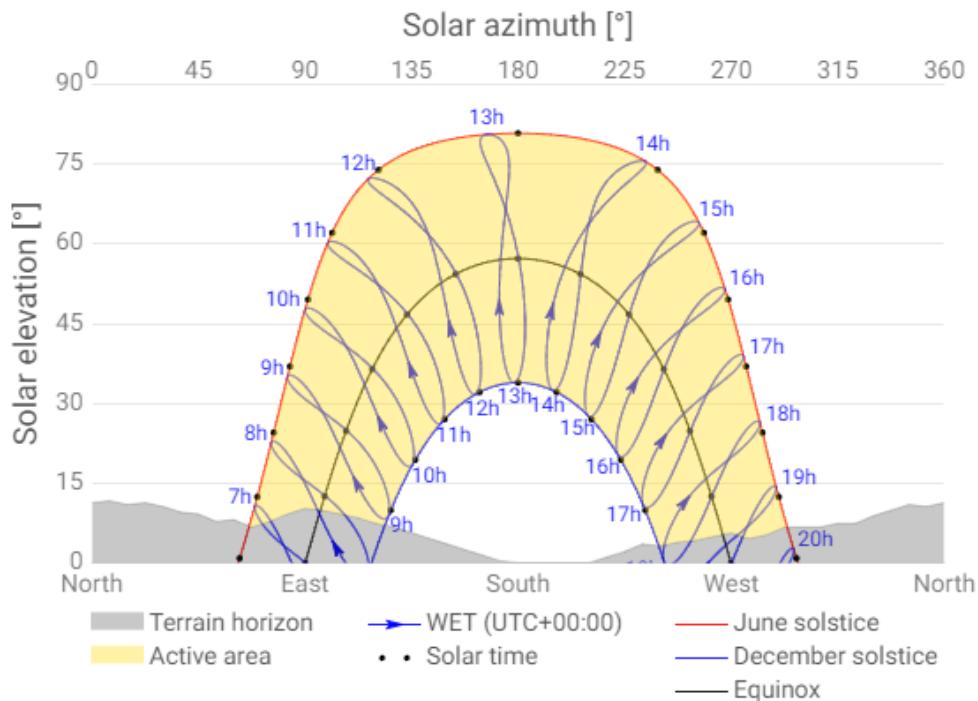


Figura 3.9: Representación de la trayectoria solar percibida en la localización de la ciudad de Funchal [83]

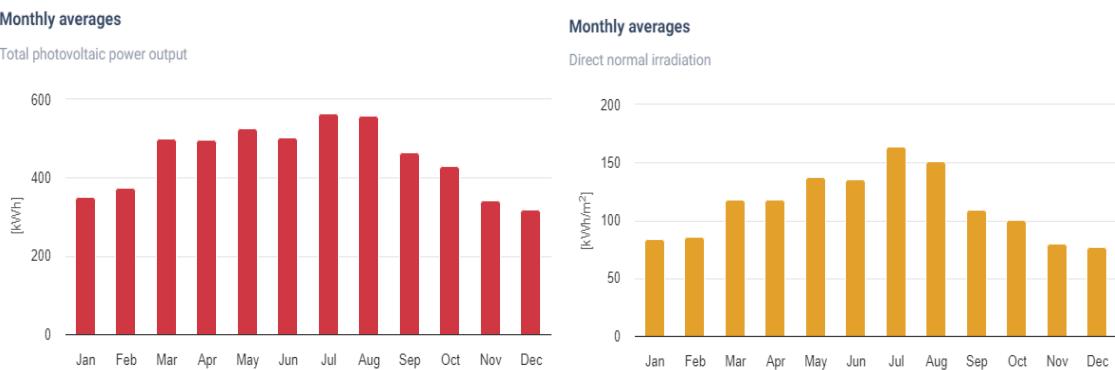


Figura 3.10: Comparación de valores totales mensuales de producción energética fotovoltaica (PVOOUT) respecto a la radiación directa normal (DNI) [83]

De la misma forma, se representan en la Figura 3.11 los perfiles diarios propromedios de radiación y generación fotovoltaica en función del mes del año, en los cuales se encuentran valores máximos en las horas centrales de los meses de julio y agosto.

Average hourly profiles

Total photovoltaic power output [kWh]

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
0 - 1												
1 - 2												
2 - 3												
3 - 4												
4 - 5												
5 - 6				0.002	0.030	0.043	0.022	0.002				
6 - 7			0.021	0.196	0.342	0.338	0.290	0.227	0.132	0.026	0.002	
7 - 8	0.036	0.164	0.512	0.831	0.965	0.942	0.917	0.871	0.816	0.649	0.352	0.118
8 - 9	0.755	0.911	1.201	1.414	1.486	1.479	1.538	1.525	1.437	1.263	0.995	0.758
9 - 10	1.307	1.469	1.710	1.826	1.812	1.760	1.911	1.965	1.831	1.703	1.455	1.247
10 - 11	1.636	1.824	2.093	2.066	2.007	1.960	2.153	2.209	2.059	2.025	1.706	1.525
11 - 12	1.773	1.982	2.259	2.132	2.078	2.012	2.256	2.306	2.074	1.989	1.734	1.602
12 - 13	1.695	1.926	2.176	2.085	2.040	1.938	2.211	2.235	1.948	1.851	1.616	1.554
13 - 14	1.511	1.731	2.013	1.919	1.947	1.901	2.107	2.098	1.755	1.638	1.445	1.379
14 - 15	1.306	1.527	1.767	1.692	1.703	1.687	1.866	1.847	1.521	1.366	1.181	1.153
15 - 16	0.929	1.139	1.365	1.308	1.324	1.330	1.478	1.432	1.132	0.929	0.765	0.742
16 - 17	0.349	0.614	0.820	0.816	0.845	0.873	0.964	0.912	0.652	0.387	0.208	0.200
17 - 18		0.069	0.217	0.296	0.356	0.406	0.443	0.367	0.152	0.012		
18 - 19				0.014	0.049	0.081	0.085	0.026				
19 - 20												
20 - 21												
21 - 22												
22 - 23												
23 - 24												
Sum	11	13	16	17	17	17	18	18	16	14	11	10

Average hourly profiles

Direct normal irradiation [Wh/m²]

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
0 - 1												
1 - 2												
2 - 3												
3 - 4												
4 - 5												
5 - 6				48	63	48						
6 - 7		136	325	352	340	242	75					
7 - 8	65	231	382	430	451	476	446	385	292	157	50	
8 - 9	293	325	385	423	443	458	521	505	438	379	315	272
9 - 10	364	383	418	415	418	411	493	502	421	403	358	337
10 - 11	374	392	437	400	399	399	484	487	405	424	353	342
11 - 12	363	382	441	384	396	383	482	476	377	373	326	320
12 - 13	333	357	410	368	386	367	471	455	344	338	298	296
13 - 14	294	328	389	347	381	378	467	438	320	312	279	277
14 - 15	284	311	366	334	361	362	442	419	304	296	261	266
15 - 16	258	274	331	304	324	328	398	370	263	249	230	228
16 - 17	135	203	275	253	269	281	331	307	219	158	97	92
17 - 18	32	121	167	201	225	248	216	92	8			
18 - 19			18	47	62	80	29					
19 - 20												
20 - 21												
21 - 22												
22 - 23												
23 - 24												
Sum	2,699	3,053	3,802	3,931	4,428	4,519	5,281	4,892	3,642	3,231	2,674	2,480

Figura 3.11: Perfiles promedios de potencial de producción energética fotovoltaica (PVOUT) y de radiación directa normal (DNI) [83]

Por lo tanto, tomando en consideración los parámetros configurados y los resultados obtenidos, se puede cuantificar finalmente, que la instalación de un sistema fotovoltaico en un edificio residencial de Funchal proporcionaría en un año una generación de energía promedia igual a 5433kWh y un valor acumulado de DNI equivalente a 1719,4 kWh/m².

3.2.2. Creación del dataset de producción (*PVWatts*)

Una vez que se han concretado a modo de análisis las características geográficas, climáticas y energéticas de la localización seleccionada en la ciudad de Funchal, se procederá a realizar el paso dedicado a la obtención de los datos de producción fotovoltaica por medio de la simulación. Para ello, se hará uso de la herramienta online *PVWatts*³ [88], proporcionada por el *National Renewable Energy Laboratory* (NREL), que se trata del laboratorio nacional del Departamento de Energía de Estados Unidos.

De la misma forma que se ha operado con la plataforma *Global Solar Atlas* (ver Sección 3.2.1), en el caso de *PVWatts*, también se requiere indicar a la entrada las características relativas a la ubicación seleccionada y a la configuración del sistema de paneles fotovoltaicos que se desea.

En cuanto a la primera, la herramienta *PVWatts*, a diferencia de *Global Solar Atlas*, provee un mapa mundial que se rige por celdas, en función de la localización de las bases de datos del NREL. En el caso de este TFM, se cuenta con la fortuna de que se dispone de una de estas bases en la misma ciudad de Funchal, concretamente en las coordenadas 32,65°N, 16,92°W. En este contexto, es preciso comentar que esta característica se tuvo en cuenta para definir la localización exacta de instalación del sistema fotovoltaico en la plataforma *Global Solar Atlas* (ver Sección 3.2.1). Por lo tanto, no se requieren ajustes adicionales en estos términos.

3.2.2.1. Configuración de los parámetros de entrada

Con el objetivo de conseguir a la salida de la simulación unos datos que sean coherentes y acordes al análisis realizado anteriormente en la Sección 3.2.1.4, es imprescindible replicar y ajustar de la forma más precisa posible los valores de los parámetros de entrada de la herramienta:

³<https://pvwatts.nrel.gov/pvwatts.php>

- Tamaño del sistema fotovoltaico y capacidad de instalación: Por defecto, *PVWatts* determina una capacidad de 4kWp, por lo que coincide directamente con la definida en *Global Solar Atlas*. Además, se especifica la relación existente entre este valor y el tamaño del sistema de la siguiente manera:

$$\frac{4 \text{ kW}}{1 \text{ kW/m}^2 \times 0,16} = 25 \text{ m}^2$$

Donde se representa que, para obtener una eficiencia del 16% del sistema (eficiencia por defecto), se requiere un área de 25 m² para los módulos solares. Es decir, este valor no representa el área total del sistema fotovoltaico, ya que para ello habría que añadir el cálculo del espacio que se necesita establecer entre cada uno de los paneles DC o el tamaño de los inversores AC que se instalarían.

- Índice de cobertura del suelo: Se define como la relación entre el área de superficie del módulo y el área del suelo (tejado en el caso de una vivienda) ocupado en total. El valor predeterminado es de 0,4, lo que supone que, teniendo un área efectiva de 25 m², se necesitará un área total de 62,5 m².
- Ratio DC/AC: Haciendo referencia al tamaño de los inversores AC, es preciso exponer que estos limitan la salida de la energía del conjunto para que haya un correcto funcionamiento. La herramienta cuantifica por defecto para la localización, una relación 1,2 entre el tamaño de array de paneles y los inversores de corriente alterna. Por lo tanto, en este caso, estos contarán con una capacidad teórica de 3,33kW. Si se pretendiera realizar una instalación a gran escala se necesitarían ratios de hasta 1,5.
- Azimut: De la misma forma que en la configuración en la plataforma *Global Solar Atlas*, se define un azimut de 180°, orientando los paneles hacia el sur.
- Inclinación: *Global Solar Atlas*, en el caso de localizar el sistema fotovoltaico en la ciudad de Funchal, establecía como ángulo óptimo de inclinación un valor de 26°.
- Tipo de módulos: Se selecciona el tipo estándar, basado en silicio cristalino y con una cobertura de vidrio con revestimiento antireflectante. Aproximadamente, cuenta con una eficiencia nominal del 19%.
- Tipo de array: Se especifica de forma simplificada un tipo de array fijo, que no siga el movimiento del sol mediante ejes de rotación, sino que mantenga sus valores de inclinación y de azimut configurados. En la Figura 3.12, se describe el funcionamiento de cada tipo de array.

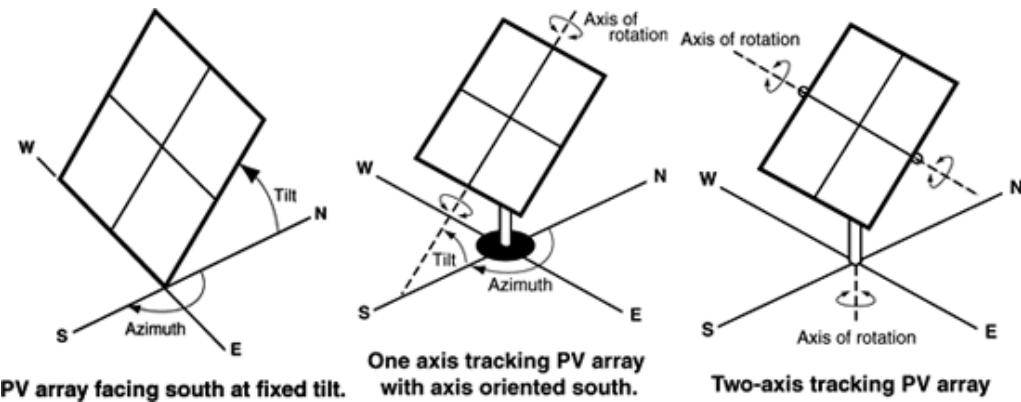


Figura 3.12: Representación de las diferentes opciones de array que permite configurar la herramienta *PVWatts* [88]

3.2.2.2. Configuración de las pérdidas del sistema real

Una de las ventajas de uso que permite la herramienta *PVWatts* es la posibilidad de estimar las pérdidas totales que tendría el sistema en la realidad. Esto es importante para que los resultados de la simulación que se va a realizar sean lo más parecidos a los que se obtendrían mediante la medición del sistema real. Para ello, se deben configurar cada una de las pérdidas parciales de la siguiente manera:

- Suciedad: Debido a la escasez de precipitaciones y a la presencia de calima que se experimenta en la isla, se determina un valor de un 2%.
- Sombreado: Se reduce la radiación solar que incide en los paneles si se producen sombras por objetos cercanos o por la propia de los módulos si se colocan en fila. Es decir, estos crean sombras sobre los de la fila adyacente. El cálculo se realiza a partir de la gráfica expuesta en la Figura 3.13. Por lo tanto, como se ha definido un tipo de array fijo con una inclinación de 26° y un valor de índice de cobertura del suelo de 0,4, se determina aproximadamente un valor de pérdidas por sombra igual a un 3%.
- Nieve: Teniendo en cuenta las características climáticas de la ciudad de Funchal, debido a su localización en una zona subtropical, se determinan pérdidas del 0%.
- Discordancia: Son pérdidas eléctricas debido a las diferencias que se producen por imperfecciones en la fabricación de los módulos. Esto tiene como consecuencia que cada uno de ellos presente características I-V que varíen ligeramente entre sí. La herramienta proporciona un valor predeterminado del 2%.

- Cableado y conexiones: Pérdidas resistivas en el cableado y en las conexiones entre módulos, inversores y otros elementos del sistema con un valor por defecto del 2% y del 0,5%, respectivamente.
- Degradación inducida por la luz: Se trata del efecto de reducción de potencia que se produce durante los primeros meses de funcionamiento del sistema fotovoltaico. El valor predeterminado es 1,5%.
- Precisión del fabricante: Son pérdidas, generalmente bajas, que vienen definidas por la precisión del fabricante en el proceso de estudio de eficiencia de los paneles. Se determina un 1%.
- Disponibilidad: Se añaden las pérdidas que se producen a partir de los mantenimientos que se requieren a lo largo del tiempo y que causan paradas en el funcionamiento del sistema. Se provee un valor por defecto del 3%.

Finalmente, las pérdidas totales del sistema se pueden cuantificar a partir de la siguiente expresión:

$$Losses = (1 - 0,02) \times (1 - 0,03) \times (1 - 0,02) \times (1 - 0,02) \times (1 - 0,005) \times (1 - 0,015) \times (1 - 0,01) \times (1 - 0,03)$$

$$100\% \times (1 - Losses) = 14,08\%$$

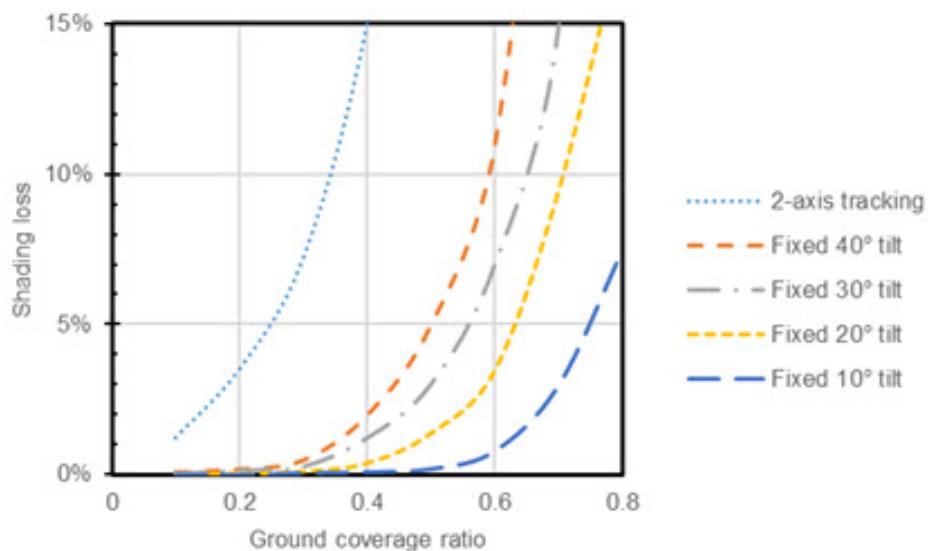


Figura 3.13: Gráfica para el cálculo de las pérdidas por sombreado en la herramienta PVWatts [88]

3.2.2.3. Análisis de resultados de la simulación

Una vez se ha concluido con la configuración necesaria, se procede a ejecutar la simulación en la herramienta y a adquirir a la salida los resultados de la misma. Se obtienen dos datasets que replican un año completo de mediciones: uno con los valores totales mensuales (ver Tabla 3.9) y otro, con un conjunto de muestras que proporcionan información de cada hora (ver Tabla 3.10).

Como se puede visualizar, en ambas tablas se proporcionan los valores de potencia DC y AC, que corresponden a la salida del módulo solar (*DC Array Output*) y de los inversores (*AC System Output*), respectivamente. En el caso de la Tabla (ver Tabla 3.10), se incluye de forma adicional toda la información climática y ambiental de cada instante.

Campo	Descripción	Unidades
<i>Month</i>	Mes de la muestra	-
<i>Daily POA Irradiance</i>	Índice de radiación en el plano del array	kWh/m ² /day
<i>DC Array Output</i>	Potencia de salida DC del array	kWh
<i>AC System Output</i>	Potencia de salida AC del sistema	kWh

Tabla 3.9: Dataset con valores totales mensuales [88]

Campo	Descripción	Unidades
<i>Month</i>	Mes de la muestra	-
<i>Day</i>	Día de la muestra	-
<i>Hour</i>	Hora de la muestra	-
<i>Diffuse Irradiance</i>	Índice de radiación difusa (DIF)	W/m ²
<i>Plane of Array Irradiance</i>	Índice de radiación en el plano del array (POA)	W/m ²
<i>Ambient Temperature</i>	Temperatura ambiente	C
<i>Wind Speed</i>	Velocidad del viento	m/s
<i>Albedo</i>	Índice de reflectividad de la superficie	-
<i>Cell Temperature</i>	Temperatura de las células solares	C
<i>DC Array Output</i>	Potencia de salida DC del array	W
<i>AC System Output</i>	Potencia de salida AC del sistema	W

Tabla 3.10: Dataset con muestras adquiridas por hora [88]

Teniendo en cuenta las tablas anteriores, en este paso será imprescindible comprobar que los datos finales obtenidos presentan coherencia respecto al estudio previo y que encajan además, con el análisis de los resultados de la plataforma *Global Solar Atlas*, realizado en la Sección 3.2.1.4.

Por ello, en primera instancia, se va a proceder a poner el foco en los valores de radiación directa normal DNI. La herramienta *PVWatts*, como se puede ver en la Tabla 3.9, no proporciona directamente información mensual sobre este parámetro, sino sobre el índice de radiación en el plano del array (del inglés *Plane of Array Irradiance* (POA)), que se calcula a partir de la expresión [89]:

$$POA = DNI \cdot \cos(\theta) \quad (3.2)$$

Donde:

θ es el ángulo de inclinación configurado, que en este caso sería de 26°.

Por lo tanto, se pueden calcular los valores de DNI a partir de los de POA dados por el dataset. Mediante la librería *matplotlib* de *Python*, se representa la Figura 3.14 para visualizar los valores totales mensuales, tanto de producción energética, como de radiación directa normal DNI. El valor máximo energético se alcanza en el mes de julio, con un valor de 517,8kWh, ocurriendo de la misma forma para el valor promedio máximo de radiación, cuyo alcance es de 181,1kWh/m² para este mes. Por tanto, volviendo a la Sección 3.2.1.4 y, específicamente a la 3.10, se puede determinar que los valores obtenidos en la simulación son muy precisos respecto al estudio previo.

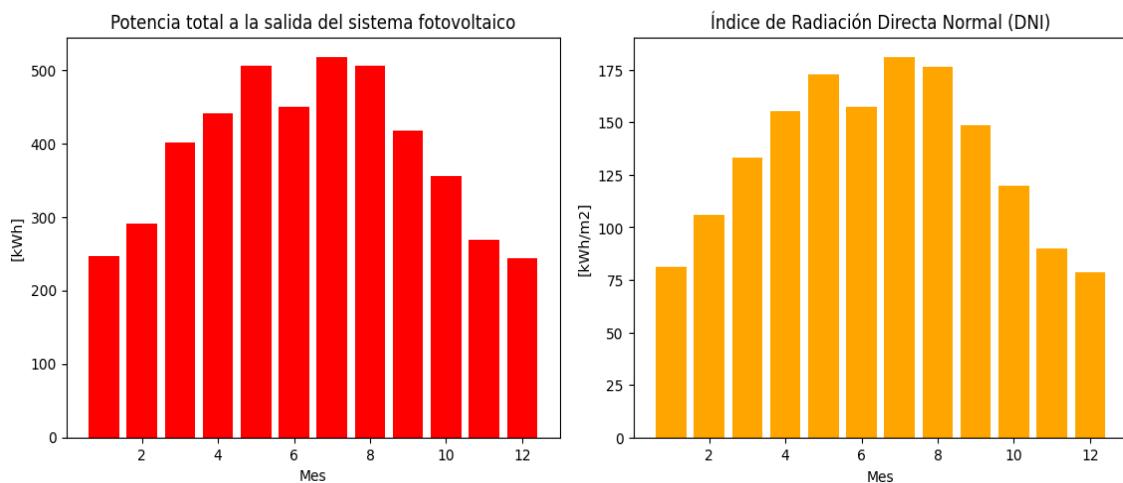


Figura 3.14: Comparación de valores totales mensuales de producción energética fotovoltaica (PVOUT) respecto a la radiación directa normal (DNI) a partir de los resultados de la simulación

A parte de las gráficas realizadas, se expone la Tabla 3.11 para llevar a cabo de una forma correcta la comparación entre los valores estudiados en *Global Solar Atlas* y los resultantes de la simulación en *PVWatts*. Como se puede ver, existe un menor error entre los valores respectivos a la radiación que entre los valores de potencia producida. Las diferencias percibidas provienen principalmente por las pérdidas configuradas, ya que al final los resultados de la simulación son mucho más cercanos al caso real que los percibidos teóricamente.

Por lo tanto, a través del sumatorio de todos los valores mensuales de producción energética, se puede cuantificar finalmente, que un sistema fotovoltaico real localizado en la ciudad de Funchal, proveería aproximadamente un total de 4417kWh al año. Este valor representa un 81,3% del obtenido teóricamente (5433kWh) en la Sección 3.2.1.4 y viene justificado en gran parte por las pérdidas introducidas. Por otro lado, en el caso del DNI, con la herramienta *PVWatts* se obtiene un total anual equivalente a 1598,8 kWh/m², suponiendo un 92,9% del valor teórico (1719,4 kWh/m²).

Es preciso tener en cuenta las diferencias que presentan las herramientas *Global Solar Atlas* y *PVWatts* en sus algoritmos de cálculo, ya que cada una de ellas asignará pesos diferentes a los parámetros de entrada, incidiendo parcialmente en los resultados. No obstante, a partir de los valores anteriores y de la Tabla 3.11, se puede determinar de forma concluyente que la simulación realizada es coherente con el estudio previo, cumpliendo así el objetivo principal de esta Sección.

Mes	Producción energética fotovoltaica (PVOUT) [kWh] Global Solar Atlas	PVWatts	Índice de Radiación Directa Normal (DNI) [kWhm ⁻²] Global Solar Atlas	PVWatts
Enero	350,2	247,1	83,7	81,1
Febrero	374,0	291,4	85,5	105,7
Marzo	500,8	401,9	117,9	132,8
Abril	497,9	441,3	117,9	155,0
Mayo	526,5	505,9	137,3	172,7
Junio	502,5	449,6	135,6	157,4
Julio	565,5	517,8	163,7	181,1
Agosto	558,6	505,5	151,6	176,1
Septiembre	465,3	417,4	109,3	148,7
Octubre	429,0	356,3	100,2	119,5
Noviembre	343,7	268,8	80,2	90,1
Diciembre	318,6	243,4	76,9	78,6

Tabla 3.11: Tabla de comparación de los valores totales mensuales de producción energética fotovoltaica (PVOUT) y de radiación directa normal (DNI) de las dos herramientas (valores teóricos y simulados)

3.3. Procesado de los datos

En las secciones anteriores, se ha detallado el proceso de búsqueda y recolección de datos energéticos en entornos residenciales a partir de las fuentes públicas disponibles. Este proceso ha venido acompañado por un estudio y análisis en profundidad para exponer de forma justificada las razones de la selección de los conjuntos de datos que servirán como base de este TFM.

También, se ha llevado a cabo una investigación enfocada en dos herramientas de extracción de información geográfica, climática y energética. El fin de este proceso ha sido incorporar una nueva fuente de datos con parámetros relacionados con la producción de electricidad, la cual se ha construido a partir de la simulación de un caso real.

Entonces, se puede expresar que la ejecución de los procesos anteriores proporcionará múltiples ficheros de datos que soporten extensas cantidades de información. En consecuencia, estos ficheros se caracterizarán por su gran tamaño y su dificultad de manejo y empleo. Por ello, se introduce esta Sección con el fin principal de realizar un procesamiento exhaustivo y reducir los conjuntos de datos únicamente a las muestras que aporten información de utilidad para llevar a cabo el desarrollo. Teniendo en cuenta el objetivo definido, se va a estructurar esta Sección en diferentes fases, en función de cada uno de los pasos que se deberán realizar para procesar los datos.

Antes de comenzar a describir este procesamiento, es imprescindible destacar que este vendrá determinado principalmente por la creación y diseño de código programado en *Python* en diferentes *notebooks* de *Jupyter*, ya que es el entorno más adecuado en el ámbito de la ciencia de datos y de la aplicación de técnicas de ML y DL. El conjunto de ficheros enfocados al preprocesado de datos se encuentran disponibles en el repositorio⁴ de GitHub dedicado a este proyecto y su empleo aportará la ventaja de poder depurar y comprobar paso a paso que se ejecutan de forma correcta cada una de las acciones requeridas en esta etapa. Cabe destacar el uso de algunas librerías que serán imprescindibles para el manejo de los datos, como son *pandas*, *csv* o *numpy*, entre otras. Adicionalmente, para la representación de las diferentes gráficas de la Sección se hará uso de los módulos *matplotlib* y *seaborn*.

⁴<https://github.com/PaulaBartolomeMora/TFM/tree/main/SustData>

3.3.1. Procesado de datos de consumo

3.3.1.1. Análisis de la situación inicial y primeros pasos

El dataset seleccionado para este TFM, *SustDataED*, se detallaba en la Sección 3.1.1 como un conjunto de datos que abría multitud de posibilidades de implementación. Se puede expresar que esto venía justificado por el hecho de abarcar un extenso lapso temporal de medidas, correspondientes a un gran número de viviendas. Además, estas medidas se caracterizaban por ser adquiridas a una frecuencia de muestreo de un minuto, lo que aportaba una buena resolución para analizar el comportamiento eléctrico de todos los usuarios implicados. Como también se exponía en la Sección 3.1.1, especialmente en la Figura 3.1, el dataset *SustDataED* se estructuraba en cuatro despliegues diferentes, cada uno con diferentes rangos temporales de medición y con un conjunto de viviendas determinado.

Poniendo el enfoque en las medidas de consumo, esto supone que se termine coleccionando en conjunto hasta un total de 24.512.181 muestras, correspondientes a 1144 días de medición. A modo de síntesis, se aporta la Tabla 5.1 con la información respectiva a cada despliegue [78].

Por lo tanto, en este paso, enfocado al análisis de la situación inicial del conjunto, es preciso tener el conocimiento de los ficheros de datos de consumo de los que se va a partir. Por este motivo, se añade de forma adicional la Tabla 3.13, la cual expone la información relativa a cada despliegue y a los tamaños de cada uno de los ficheros proporcionados en el dataset.

Despliegue	Muestras	Días	Fecha de inicio	Fecha de fin
1	3.474.557	123	10/07/2010	10/11/2010
2	12.481.536	504	25/11/2010	20/04/2012
3	5.671.576	298	01/08/2012	25/05/2013
4	2.884.512	219	31/07/2013	10/03/2014
TOTAL	24.512.181	1144		

Tabla 3.12: Resumen de los datos de consumo del dataset *SustDataED* [78]

Como se puede comprobar, a priori la mayoría no precisan de un volumen manejable, por lo que la primera acción a realizar será dividir los mismos en nuevos ficheros que supongan un tamaño menor del dado. Esto sobre todo será importante a la hora de trabajar con el repositorio de GitHub, puesto que existe el factor limitante de que el tamaño máximo permitido por archivo es de 100MB.

No obstante, antes de dividir los ficheros, se debe volver a la Tabla que hace referencia a las medidas de consumo energético (ver Tabla 3.2) para observar que estos ficheros contienen una columna dedicada a la marca de tiempo (*timestamp*) del instante en el que se adquirió la medida.

Despliegue	Fichero de muestras	Tamaño (MB)
1	power_samples_d1_1	86.38
	power_samples_d1_2	69.99
2	power_samples_d2_1	289.81
	power_samples_d2_2	265.13
	power_samples_d2_3	283.27
	power_samples_d2_4	311.89
3	power_samples_d3_1	259.92
	power_samples_d3_2	233.69
	power_samples_d3_3	272.23
4	power_samples_d4_1	250.12
	power_samples_d4_2	252.84

Tabla 3.13: Resumen de las características de los ficheros de los despliegues de *SustDataED*

A modo de simplificar esta información temporal, se crea un nuevo *notebook* de *Jupyter*, denominado como *preprocessing_nodes.ipynb*, que va a importar la librería *time* para añadir tres nuevas columnas al dataset que separen los datos de la fecha (columna *datetime*), el instante exacto de la muestra (hora, minuto y segundo) (columna *hour*) y la hora como un valor entero (columna *H*). En el caso de esta última columna, su utilidad vendrá descrita por los requerimientos del Apartado posterior (ver Sección 3.3.1.3).

Código 3.1: Formato del timestamp

```

1 df['tmstp'] = pd.to_datetime(df['tmstp']) # Conversión de la columna del \textit{dataframe} a formato ←
    ↪ datetime
2 df['datetime'] = df['tmstp'].dt.strftime('%Y-%m-%d') # Formateo de la fecha (año, mes, día)
3 df['hour'] = df['tmstp'].dt.strftime('%H:%M:%S') # Formateo del instante (hora, minuto, segundo)
4 df['H'] = df['tmstp'].dt.strftime('%H').astype(int) # Formateo de la hora como entero

```

De forma adicional, en este *notebook* se comprueba también, si existen filas con valores *NaN* que puedan perjudicar al análisis de los datos. Como ventaja, el dataset *SustDataED* aporta la columna *miss_flag* (ver Tabla 3.2) para determinar si para cierto instante temporal se ha producido la adquisición de forma incorrecta o con valores vacíos. Por tanto, la acción a desarrollar será comprobar en todo el *dataframe* cuándo este valor se encuentra activo y, en ese caso, eliminar la fila respectiva.

Código 3.2: Eliminación de valores NaN

```

1 df.isnull().any() # Comprobación previa de existencia de valores NaN en el \textit{dataframe}
2 df = df[df['miss_flag'] != 1] # Aplicación de filtro al \textit{dataframe}
3 df = df.dropna(thresh=len(df.columns) - 15 + 1) # Eliminación de columnas con todos los elementos ↪
    ↪ vacíos
4 df.isnull().any() # Comprobación posterior de existencia de valores NaN en el \textit{dataframe}

```

A modo de resumen, se puede determinar entonces, que se seguirá la siguiente secuencia de pasos en esta primera fase: crear las nuevas columnas temporales, eliminar las filas que estén completamente vacías y dividir los ficheros iniciales para ajustar su tamaño a un máximo de 100MB. Es obligatorio seguir este orden, puesto que no sería eficiente extraer los fragmentos del conjunto de datos y, después, añadir información nueva. En otros términos, de esta forma se estaría superando el tamaño máximo ajustado en los nuevos ficheros de salida.

Volviendo al proceso de división de los ficheros, se crea otro *notebook* de *Jupyter*, denominado como *split.ipynb*. En primer lugar, se importa al mismo la librería de *Python csv*, enfocada al tratamiento de archivos con formato .csv. Después, se escribe una función que, a partir de un fichero de entrada, itere a través de sus filas de datos hasta llegar al volumen máximo determinado para obtener a la salida el nuevo archivo en la ruta indicada. Es importante indicar en la función que, para cada nueva fragmentación de los datos, se incluya el encabezado con los nombres de cada uno de los campos, puesto que suponen ficheros diferentes y, de otro modo, no vendrían definidas correctamente sus columnas.

Código 3.3: Fragmentación de ficheros iniciales

```

1 with open(input_file, 'r') as csv: # Apertura del fichero de entrada
2     reader = csv.reader(csv) # Definición del reader
3     header = next(reader)
4
5     for row in reader: # Iteración de las filas del fichero
6         if current_row % split_size == 0: # Condición de máximo de filas
7             if output_file: # Cierre del fichero de salida
8                 output_file.close()
9             split = f'dataset/split_{current_split}.csv'
10            output_name = input_name + split
11            output_file = open(output_name, 'w', newline='') # Apertura del fichero de salida
12            writer = csv.writer(output_file) # Definición del writer
13            writer.writerow(header) # Escritura del encabezado
14            current_split += 1
15            writer.writerow(row) # Escritura de fila
16            current_row += 1
17
18 if output_file: # Cierre del fichero de salida
19     output_file.close()

```

Por tanto, tras realizar este procedimiento, se logra obtener un total de 28 nuevos ficheros en formato .csv de un tamaño reducido, permitiéndose así, una mejor gestión de los datos que contienen los mismos.

3.3.1.2. Síntesis de la información

La presente Sección se dedicará principalmente a la reducción de la información dada por *SustDataED*. Como se ha comentado anteriormente, el hecho de que las muestras se hayan adquirido a una frecuencia de un minuto, proporciona un gran volumen de datos que debería sintetizarse. En el caso de este TFM, con el fin de reducir el número de filas del dataset, se ha valorado como opción más apropiada realizar una cuantificación del promedio de las mediciones para cada hora. En otros términos, a partir de las muestras iniciales obtenidas cada minuto, se generaría un nuevo conjunto de medidas promediadas cada hora, resultando en una reducción de los datos en un factor 60.

En primera instancia, para simplificar la programación, se crea un primer *notebook* básico que realice el procedimiento de reducción de datos para uno de los ficheros fragmentados que se han obtenido a partir de *split.ipynb*. Este nuevo *notebook* se denomina como *datasamples_onenode.ipynb* y aplica el promedio únicamente a las medidas correspondientes a una hora y día determinados de una vivienda en particular. Estos valores son determinados como parámetros de entrada y, en el caso de la hora, es preciso hacer uso de la columna *H*, definida en la Sección anterior (ver Sección 3.3.1.1) para determinar la hora a la que pertenece la medida como un valor entero.

Por tanto, una vez definidos los valores de los parámetros de entrada, se aplica un filtro al *dataframe* a partir de los mismos y se comprueba que se ha realizado el procedimiento de forma correcta mediante la observación del número de filas que se extraen, que debería ser 60. Para llevar a cabo el cálculo del promedio, es preciso especificar las columnas sobre las que se va a aplicar la operación, que son las que contienen todos los valores que hacen referencia a los parámetros eléctricos. Por tanto, a partir de las 60 filas filtradas y, tomando en consideración las columnas comentadas, se puede calcular ya el valor promedio de cada uno de los parámetros eléctricos.

El resultado de este proceso supone una única fila con todos los valores de los parámetros promediados de una hora. En la Figura 3.15, se muestra como ejemplo la salida que se obtiene tras aplicar el promedio a las medidas adquiridas en la vivienda con identificador 25 en el instante temporal correspondiente al 8 de abril de 2013 con hora 15:00.

	Imin	Imax	Iavg	Vmin	Vmax	Vavg	Pmin	Pmax	Pavg	Qmin	Qmax	Qavg	Pfmin	Pfmax	Pfavg	iid	H	datetime
0	0.567326	1.173665	0.591531	239.589517	240.550417	240.065317	104.408123	247.581008	110.18881	-23.030628	14.021733	9.078678	0.773812	0.814176	0.791915	25	15	2013-04-08

Figura 3.15: Salida de *datasamples_onenode.ipynb* tras aplicar la operación de promedio

3.3.1.3. Automatización del proceso de síntesis de la información

Como se viene exponiendo en la Sección anterior (ver Sección 3.3.1.3), el proceso de cálculo del promedio de valores viene determinado por los parámetros de entrada que se definen y se produce una ejecución únicamente para las 60 muestras que se han adquirido a partir de un identificador de vivienda y en una hora y fecha específicas.

Esta característica supone que, si se desea aplicar el promedio a los valores de las medidas de cada hora para todo el rango temporal que abarca cada uno de los nodos, se debe automatizar el proceso. Ante esta necesidad, se crea un nuevo archivo de *Python*, denominado como *datasamples_process.py*, que incluirá todas las funciones necesarias para llevar a cabo el procedimiento de automatización. Es importante comentar que en esta Sección se pone el foco en los datos de consumo. Sin embargo, las funciones que se van a exponer a continuación, serán configuradas para ser empleadas también, en el caso de los datos relativos a la producción y al clima de *SustDataED*.

En primer lugar, *datasamples_process.py* describe una función de lectura que extrae como argumento el *path* donde se ubican los ficheros .csv a leer y, después, aplica un bucle para concatenar todos ellos en un mismo *dataframe*. Por consiguiente, se comprueba qué tipo de datos contienen los ficheros (ficheros de consumo o de producción) y se almacenan las columnas de los parámetros eléctricos con los que se va a operar en una variable X. En el caso de los ficheros de consumo, se selecciona la columna *Pavg*, que hace referencia a los valores de potencia activa media adquirida en cada instante. Esto se debe a que se trata de la información que se requiere conocer para poder establecer más adelante las cargas en el algoritmo DEN2NE. Este proceso en cuestión será detallado en la Sección 3.5.1.

En el paso de lectura se extraen también los identificadores de las viviendas que vienen contenidos en cada fichero. Finalmente, se devuelven el *dataframe* completo, la variable X y dos listas: una con los identificadores extraídos en el paso anterior y otra con todas las fechas únicas (columna *datetime*) que contiene el *dataframe*.

Código 3.4: Función de lectura de los ficheros

```

1  def read_files(files):
2      path = gl.glob(files) # Definición del path
3      dfs = []
4
5      for file in path:

```

```

6     dfs.append(pd.read_csv(file, low_memory=False)) # Lectura de cada fichero
7     print("Leyendo " + file)
8
9     df = pd.concat(dfs, ignore_index=True) # Operación de concatenación
10
11    # Comprobación de tipo de ficheros para almacenar los parámetros eléctricos y los identificadores
12    if files.startswith(files_cons): # Ficheros de consumo
13        X = df.iloc[:, 11:12]
14        iid_array = df["iid"].unique()
15    elif files == files_prod: # Ficheros de producción
16        X = df.iloc[:, 16:17]
17        iid_array = 0
18
19    X_features = X.columns.to_list()
20    X_features.append("iid")
21    X_features.append("datetime")
22    X_features.append("h")
23    datetime_array = df["datetime"].unique() # Extracción de las fechas
24    return df, X_features, iid_array, datetime_array

```

La siguiente función que se define es la relativa al cálculo del promedio de los valores. En ella, al igual que en la anterior, es preciso comprobar si los ficheros con los que se va a operar contienen datos de consumo o de producción. Se aplica una condición para ello a partir de la existencia de información sobre el identificador de la vivienda, ya que, como se comentaba en la Sección 3.1.2.4, los datos de producción de *SustDataED* se proporcionan en términos globales a todos los hogares. Esto quiere decir que no proveen información de identificador, por lo que es posible conocer el tipo de ficheros que se está tratando a partir del valor de esta columna.

Por tanto, tomando lo anterior en consideración, se aplica el filtro al *dataframe* de la misma forma que se realizaba en la Sección y se informa, a modo de depuración, sobre el progreso del proceso. El siguiente paso es calcular la media de los valores para cada hora y volcar los resultados, junto con el identificador de la vivienda y el dato de la fecha, en un nuevo *dataframe*.

Código 3.5: Función de cálculo del promedio

```

1  def extract_mean(df, IID_SAMPLE, HOUR_SAMPLE, DATETIME_SAMPLE, X_features, ↵
2      ↵ result_dict, progress):
3      # Comprobación de tipo de ficheros para aplicar el filtro
4      if IID_SAMPLE == 0: # Ficheros de producción
5          filter = df[(df["h"] == HOUR_SAMPLE) & (df["datetime"] == DATETIME_SAMPLE)]
6          desc = f"Procesando producción {DATETIME_SAMPLE} {HOUR_SAMPLE}"
7      else: # Ficheros de consumo
8          filter = df[(df["iid"] == IID_SAMPLE) & (df["h"] == HOUR_SAMPLE)
9          & (df["datetime"] == DATETIME_SAMPLE)]
10         desc = f"Procesando consumo de IID {IID_SAMPLE} {DATETIME_SAMPLE} {HOUR_SAMPLE}"

```

```

10
11 df2 = pd.DataFrame() # Creación del nuevo \textit{dataframe} para almacenar los resultados
12
13 for feature in X_features:
14     if feature == "iid":
15         df2[feature] = IID_SAMPLE
16     elif feature == "datetime":
17         df2[feature] = DATETIME_SAMPLE
18     else:
19         df2[feature] = [filter[feature].mean()] # Cálculo del promedio
20
21 result_dict[IID_SAMPLE] = df2 # Diccionario final con todos los \textit{dataframes} creados
22 progress.value += 1

```

Finalmente, se escribe la función que se va a encargar de automatizar el proceso de cálculo de todos los valores promedios mediante compresión de listas. Se produce la llamada a la función *extract_mean()* con un doble bucle que introduce iteraciones en función de todas las fechas contenidas en la lista de fechas y de todas las horas que hay en un día.

Código 3.6: Función de automatización del cálculo del promedio

```

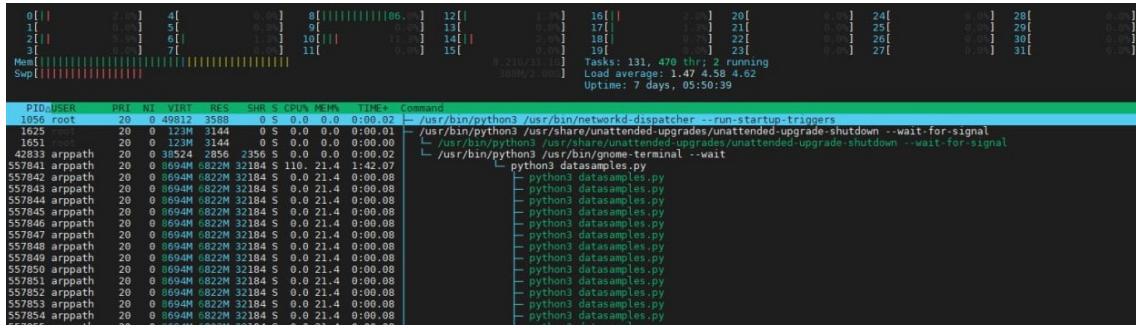
1 def process_data(iid, df_power_samples, datetime_array_ps, X_ps, result_dict, progress):
2     extraction = [
3         extract_mean(df_power_samples, iid, hour, datetime, X_ps, result_dict, progress)
4         for datetime in datetime_array_ps
5         for hour in range(24)]

```

Por lo tanto, una vez definidas las funciones anteriores, ya se permite realizar una ejecución automatizada del proceso de síntesis de los datos. En este caso, para realizar las pruebas de funcionamiento, se crea el script *run_tests.sh* y se lanza en un equipo que cuenta con un total de 32 cores. Este script posibilita la descompresión de los ficheros de datos, la instalación de las librerías de *Python* necesarias (especificadas en un archivo *requirements.txt*) y la ejecución de *datasamples_process.py*.

Se plantea en primera instancia, establecer una ejecución multihilo. Cuando se lleva a cabo el lanzamiento, se observa que, como es de esperar, se están utilizando varios cores. Sin embargo, como se puede visualizar en la Figura 3.16, el uso de *threads* no provoca en ningún momento que alguno de los cores disponibles se aproxime al 100% de su operabilidad.

Esto se debe a que los hilos no son procesos y están principalmente enfocados a la concurrencia. Es decir, el planificador del kernel establece en qué core se va a ejecutar cada una de las tareas y se producen cambios de contexto, que tienen como consecuencia, la introducción de latencias y una pérdida de eficiencia en la ejecución de las pruebas. Teniendo esto en cuenta, se toma la decisión de operar con la librería de *multiprocessing*

Figura 3.16: Lanzamiento de pruebas con *threads*

[90]. En este caso, no se van a producir desalojos, dado que la tarea sí es un proceso. Además, se establece uno para cada identificador, suponiendo que cada uno de ellos se ejecute completamente en un core. En la Figura 3.17, se muestran los cambios comentados.

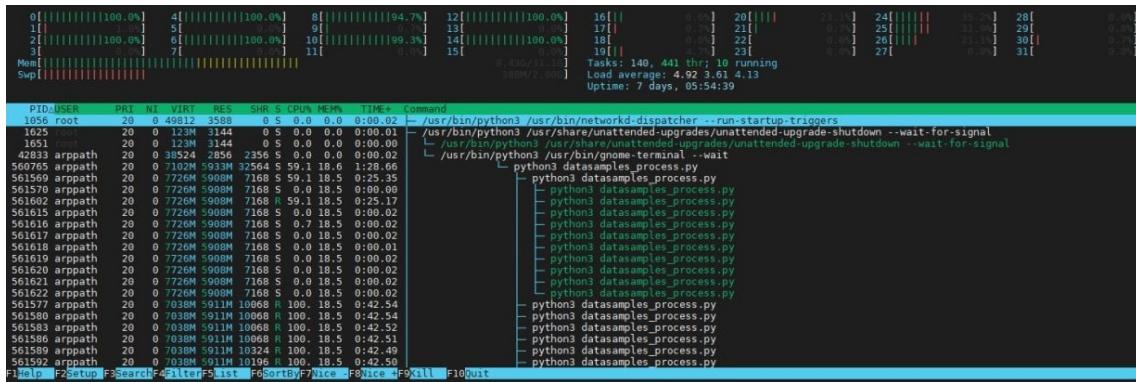


Figura 3.17: Lanzamiento de pruebas con procesos

Por lo tanto, volviendo al código, en el archivo se debe importar la librería *multiprocessing* y definir el administrador (*manager*) para crear un diccionario que será compartido por los procesos para almacenar los resultados de la ejecución. Después, se crean los procesos, llamando a la función *process_data*, expuesta anteriormente. Para la misma, se deben añadir los parámetros de entrada necesarios como argumentos e iterar para todos los identificadores de viviendas. En este momento, ya se pueden iniciar los procesos y definir el progreso de la ejecución a través de la terminal.

Para obtener a la salida los nuevos ficheros, se debe esperar a que todos los procesos finalicen. Se consigue un fichero de datos de consumo energético por vivienda definida en *SustDataED*, suponiendo en total 50. A modo organizativo, se sigue la nomenclatura *consum_x.csv*, donde x hace referencia al identificador de la vivienda a la que pertenecen

las medidas promediadas. En la Tabla 3.14 se muestran los campos que contienen estos ficheros.

Código 3.7: Creación de procesos

```

1 if __name__ == "__main__":
2     df_power_samples, X_ps, iid_array_ps, datetime_array_ps = read_files(files_cons) # Lectura de ↵
        ↵ ficheros
3
4     manager = Manager() # Creación del administrador
5     result_dict = manager.dict() # Definición del diccionario compartido de resultados
6     progress = manager.Value('i', 0) # Creación de valor para indicar el progreso
7
8     processes = [ # Creación de los procesos
9         Process(target=process_data, args=(iid, df_power_samples, datetime_array_ps, X_ps, result_dict, ↵
            ↵ progress))
10        for iid in iid_prueba
11    ]
12
13    for process in processes:
14        process.start() # Inicio de los procesos
15
16    total_tasks = len(datetime_array_ps) * len(iid_prueba) * 24 # Cálculo del número de tareas
17
18    # Creación de barra de progreso
19    with tqdm(total=total_tasks, desc="Progreso total del script") as pbar:
20        while progress.value < total_tasks:
21            pbar.update(progress.value - pbar.n)
22
23    for process in processes:
24        process.join() # Espera a la finalización de todos los procesos
25
26    for iid, result in result_dict.items(): # Almacenamiento de los resultados en los nuevos ficheros
27        result.to_csv(f"results/consum_{iid}.csv", index=False)

```

En este caso, la información de consumo, la cual venía dada a una frecuencia de muestreo de un minuto, se ha transformado a valores promediados por hora. Por tanto, se puede expresar de forma concluyente, que se logra el objetivo de reducir el tamaño de los ficheros y de sintetizar el volumen de datos a manejar, ya que a partir de ahora se va a trabajar con datos por hora.

Campo	Descripción	Unidades
<i>iid</i>	Identificador de vivienda	-
<i>datetime</i>	Fecha del valor promedio	datetime
<i>H</i>	Hora del valor promedio	-
<i>Pavg</i>	Potencia activa media	W

Tabla 3.14: Estructura de campos de cada fichero *consum_x.csv*

3.3.1.4. Selección del despliegue

Teniendo en cuenta los resultados obtenidos en la Sección anterior, el siguiente paso consiste en seleccionar las viviendas sobre las que se va a basar el desarrollo y entrenamiento de modelos de ML. En la Sección 3.1.2, se describían los distintos despliegues que componían el proceso de la adquisición y creación del dataset *SustDataED* y se representaban los rangos temporales que comprendían cada uno de ellos en la Figura 3.1.

Poniendo en consideración esta información, es imprescindible seleccionar un conjunto de viviendas cuyas medidas estén abarcadas en un mismo lapso de tiempo para poder realizar un análisis de forma precisa y entrenar correctamente los modelos de ML. En otros términos, la selección de múltiples viviendas se debe reducir a escoger el despliegue más adecuado de los cuatro en los que se estructura *SustDataED*. El objetivo es buscar un balance entre manejar un buen número de hogares y un rango temporal lo suficientemente largo. Volviendo a la Figura 3.1, se puede visualizar que los despliegues que cumplen mejor estas condiciones son el primero y el segundo.

No obstante, en este proceso de análisis y selección del conjunto de viviendas, se deben añadir también, los datos de producción energética que provee *SustDataED*. En la Figura 3.2, se muestra cómo estos datos vienen comprendidos entre el mes de octubre de 2010 y enero de 2013. Esto supone que se debe descartar el primer despliegue de viviendas, ya que este abarca un rango temporal que va desde julio de 2010 a noviembre del mismo año.

Por lo tanto, teniendo estas condiciones y características en cuenta, se justifica que el despliegue más adecuado a emplear es el segundo, el cual comprende un total de 23 viviendas (desde el identificador 1 al 23). En la Figura 3.18, se representan, mediante el empleo de la librería de *Python*, *matplotlib*, los rangos temporales que abarcan las medidas adquiridas para cada una de las viviendas participantes en el segundo despliegue.

3.3.1.5. Selección del rango temporal y procesamiento adicional

Como se puede ver en la Figura 3.18, aunque las 23 viviendas pertenecen al mismo despliegue y abarcan un lapso temporal parecido, se pueden visualizar ciertas diferencias entre las mismas. Esto supone que realmente cada fichero de datos de consumo tenga un número de muestras diferente, lo que perjudicaría al análisis y entrenamiento posterior. Una vez seleccionado el despliegue, se deben acotar los ficheros de datos de consumo a un mismo rango temporal en el se pueda trabajar de forma eficiente.

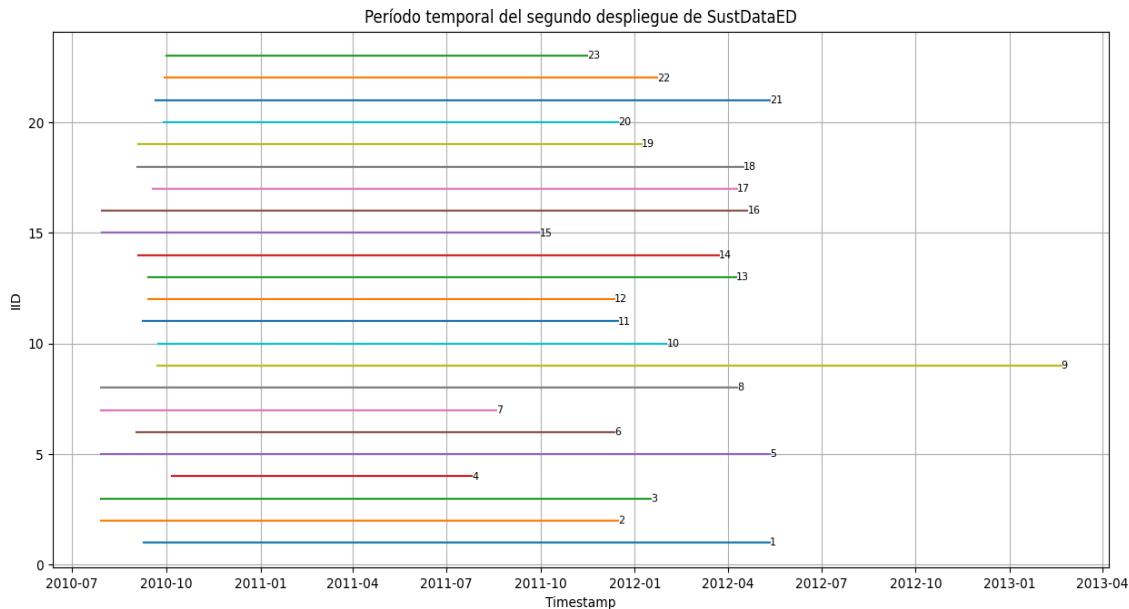


Figura 3.18: Representación del período temporal que comprende el proceso de recolección de datos para los hogares del segundo despliegue

Como se ha indicado anteriormente, los datos de producción energética se comienzan a medir desde el mes de octubre de 2010, por lo que es necesario como mínimo establecer el inicio del rango temporal en estas fechas. Entrando en el contenido de los ficheros, la fecha mínima en la que se visualiza que las 23 viviendas adquieren y almacenan medidas es el 28 de noviembre de 2010, lo que lleva a tomar la decisión de establecer la misma como inicio del rango temporal a considerar. En cuanto a la fecha final, a partir de la información dada en la Figura 3.18, se considera que la opción más eficiente es abarcar un año completo de muestras, siendo la fecha final el 28 de noviembre de 2011.

No obstante, en Secciones siguientes, se expondrá cómo el hecho de seleccionar un año de medidas será importante para calcular las cargas finales que se fijarán en el algoritmo DEN2NE (ver Secciones 3.3.3 y 3.5.1), puesto que en el caso de los datos de producción obtenidos de la herramienta *PVWatts* se comprenden valores de potencia de un año completo (ver Sección 3.3.2.2).

Código 3.8: Aplicación del rango temporal a los ficheros

```
1 rango = df[(df['datetime'] >= '2010-11-28') & (df['datetime'] < '2011-11-28')]
```

La aplicación del filtro del rango temporal a los datos se realiza en el *notebook* *consum_range.ipynb*. De forma adicional, se define el procesamiento que requieren los ficheros de consumo de aquellas viviendas que no incluyen valores en algunos instantes temporales dentro del rango indicado. En este paso, el objetivo es contar con ficheros que contengan un mismo número de medidas para simplificar su manejo. Por ello, para que los datos finales tengan la mayor precisión posible, se deben añadir las muestras que faltan en función del tipo de vivienda a la que se hace referencia.

Es decir, como se expuso en la Sección 3.1.2.3, el dataset *SustDataED* provee la Tabla 3.4 con toda la información respectiva a cada uno de los hogares monitorizados y a las características sociales de los propios inquilinos. Esta información determina principalmente, el comportamiento eléctrico que toma cada uno de los hogares y se debe tener en cuenta para añadir las muestras necesarias a los ficheros de aquellas viviendas que tienen el rango temporal incompleto.

A modo de ejemplo se muestra la Figura 3.19, donde para la vivienda con identificador 5, se podrían aproximar las muestras a las dadas por la vivienda con identificador 17. Esto se debe a que la información demográfica de las mismas es similar, ya que están habitadas por una familia de 4 integrantes, de los cuales son 2 adultos y 2 niños. Por lo tanto, se puede apreciar que se obtiene un comportamiento eléctrico parecido.

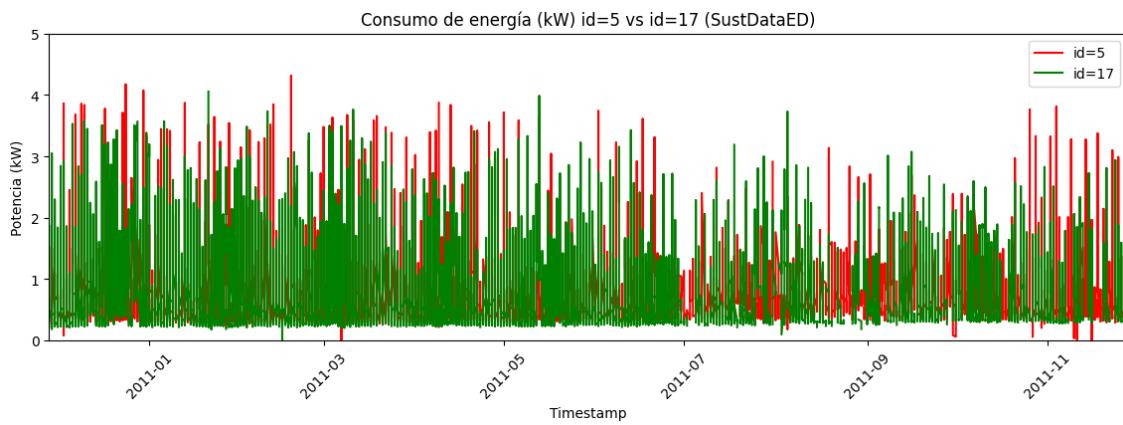


Figura 3.19: Comparación del consumo de las viviendas 5 y 17 en el rango seleccionado (*SustDataED*)

Ocurre de la misma forma en la Figura 3.20. No obstante, en este caso se representa el consumo de potencia de las viviendas con identificador 13 y 16, que se caracterizan por estar habitadas por dos inquilinos adultos, por lo que es coherente que se obtenga un menor consumo medio respecto al caso anterior.

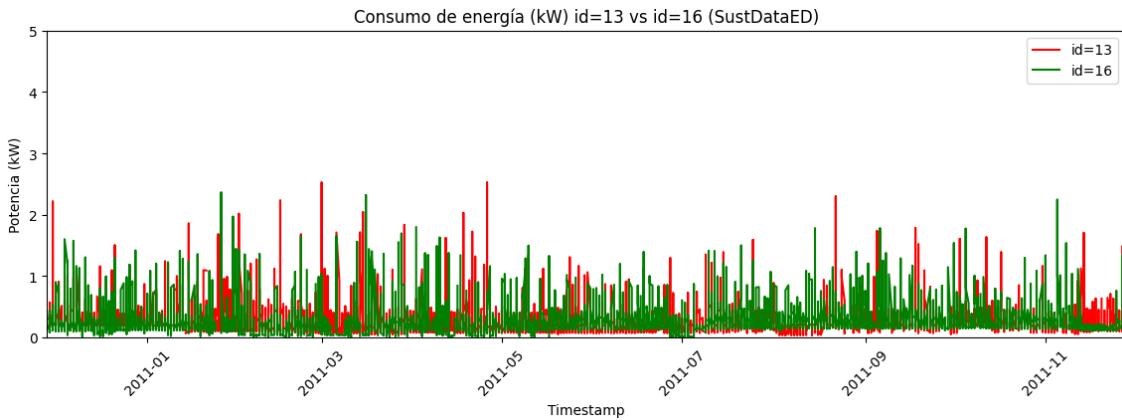


Figura 3.20: Comparación del consumo de las viviendas 13 y 16 en el rango seleccionado (*SustDataED*)

3.3.2. Procesado de datos de producción

Después de exponer el procesamiento necesario para los datos de consumo, se va a profundizar en la producción energética. Como ya se había detallado en las Secciones 3.1.3 y 3.2, referentes, respectivamente, a las conclusiones obtenidas del dataset *SustDataED* y a la introducción a las herramientas de simulación de datos de producción energética, se escribe la presente Sección con el objetivo de tomar la decisión de seleccionar la fuente de datos de producción más adecuada para el desarrollo de este TFM.

Por tanto, se debe comparar las características que presentan, tanto los datos que proporciona *SustDataED*, como los extraídos a través de la simulación en la herramienta *PVWatts*. Para ello, hay que determinar unos pasos iniciales de procesamiento en cada uno de ellos que permitan finalmente, establecer una comparación de forma correcta. En otras palabras, no sería preciso comparar ambas fuentes directamente si estas no están comprendidas en unos mismos rangos temporales o si no contienen los mismos parámetros.

3.3.2.1. Preprocesamiento de datos de *SustDataED*

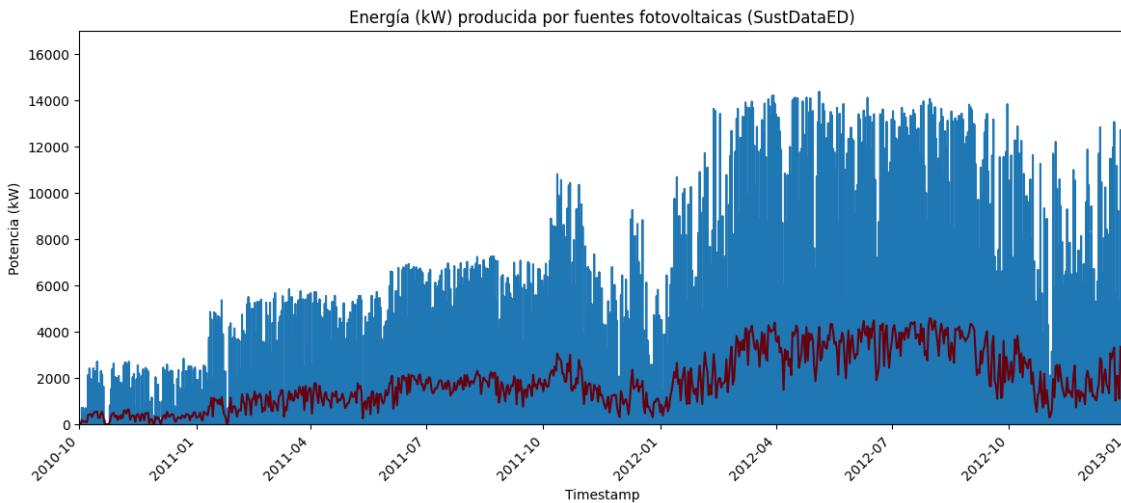
En cuanto al dataset *SustDataED*, por un lado, se realiza la combinación de los parámetros energéticos y los climáticos. Este paso viene dado por la necesidad de realizar un análisis de la correlación que existe entre la producción solar y las características ambientales que se miden en cada instante. Por ello, se crea un nuevo *notebook*, denominado como *preprocessing_production.ipynb*, en el cual hay que tener en cuenta los campos expuestos en las Tablas 3.5 y 3.7. La operación de combinación de los datos energéticos y climáticos se determina a partir de la marca de tiempo (*timestamp*) y resulta en un nuevo *dataframe* de *Pandas*, que será almacenado en un nuevo fichero en formato .csv.

Código 3.9: Combinación de datos de clima y de producción

```
1 df_merged = pd.merge(df_env, df_prod, on='timestamp') # Operación de merge
```

Aquí es necesario sintetizar los datos de producción de la misma forma que se ha realizado anteriormente para los ficheros de consumo. Como se había detallado en la Sección 3.3.1.3, se emplea el *notebook datasamples_process.ipynb* para automatizar este proceso y obtener la media de los valores de potencia. Como este TFM se engloba en un contexto de SGs, se enfoca la operación en la columna de medidas globales de producción fotovoltaica. Tras la ejecución del *notebook*, se obtiene un fichero con los campos definidos en la Tabla 3.15. Adicionalmente, se emplea la librería de *Python*, *matplotlib*, para representar gráficamente en la Figura 3.21 los valores de potencia recogidos en el fichero en órdenes de kW para simplificar el análisis.

Campo	Descripción	Unidades
<i>datetime</i>	Fecha del valor promedio	datetime
<i>H</i>	Hora del valor promedio	-
<i>solar</i>	Electricidad producida por fuentes fotovoltaicas	MWh

Tabla 3.15: Estructura de campos del fichero *mean_prod.csv***Figura 3.21:** Gráfica de energía solar producida a nivel global (*SustDataED*)

De la misma forma, teniendo en cuenta el rango temporal que se ha ajustado anteriormente para los datos de consumo (ver Sección 3.3.1.5), se comparan los valores de temperatura y de potencia medidos en las Figuras 3.22 y 3.23, respectivamente.

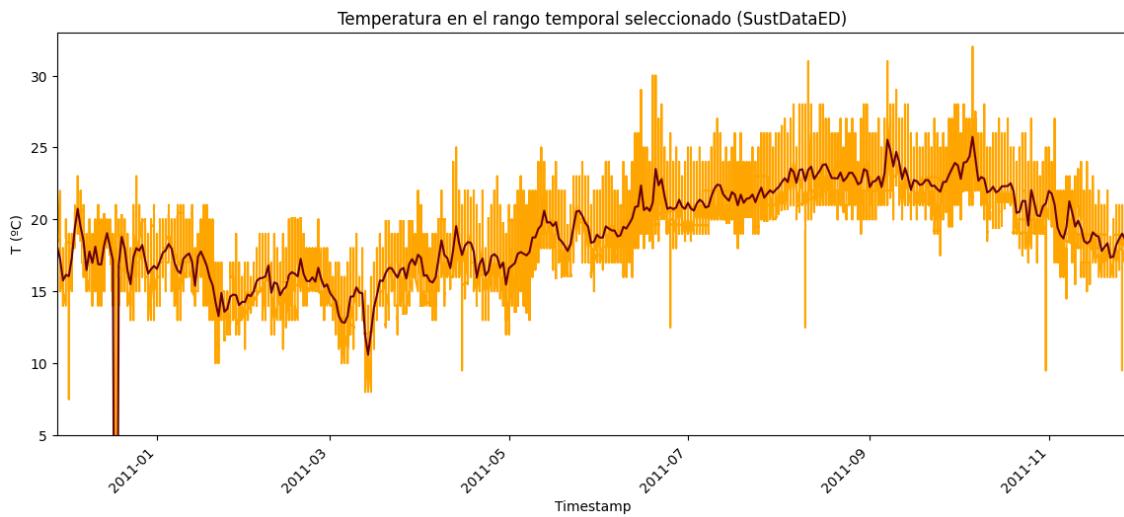


Figura 3.22: Gráfica de temperatura en el rango seleccionado (*SustDataED*)

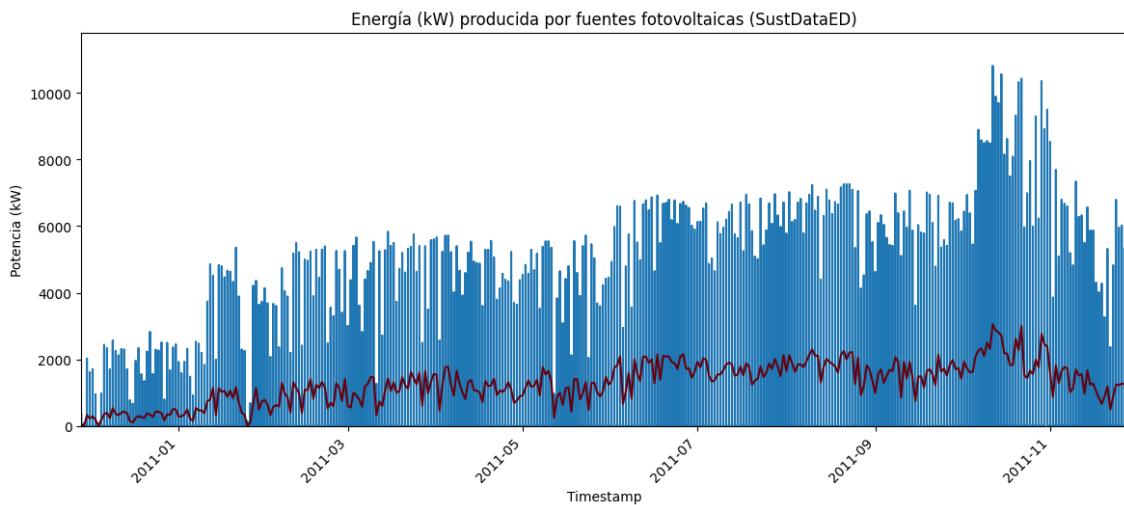


Figura 3.23: Gráfica de energía solar producida en el rango seleccionado (*SustDataED*)

A priori, se podría apreciar que los valores reflejados de generación energética presentan cierta correlación con la temperatura a partir del mes de enero de 2011 y se encuentran diferencias entre los rangos de valores que se manejan en los meses de invierno y los de verano, a medida que se incrementa la temperatura. El análisis se fija a partir de este mes porque es en este momento cuando crecen las dimensiones del sistema fotovoltaico y se produce un salto en los valores medidos, suponiendo un incremento de la potencia que se puede generar.

Por tanto, no sería correcto enfocar la comparación entre la temperatura y la potencia en los meses correspondientes al año 2010, ya que el sistema fotovoltaico no es equivalente al del año posterior. De forma adicional, se muestran de una forma más gráfica en la Figura 3.24 los valores de temperatura y potencia generada en cada instante para la primera semana de agosto.

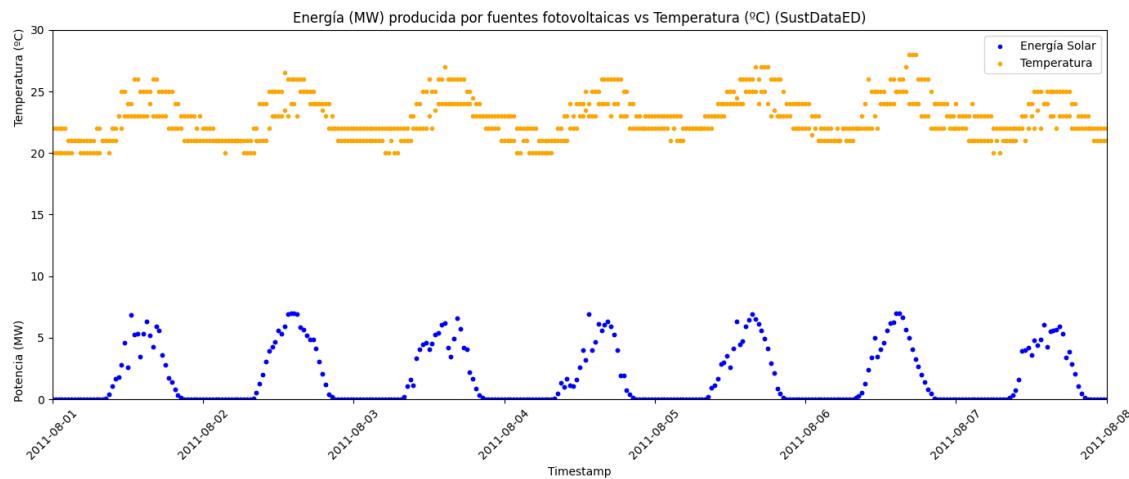


Figura 3.24: Correlación entre temperatura y energía solar producida en el mes de agosto (*SustDataED*)

No obstante, la Figura 3.24 puede dar una perspectiva equivocada sobre la relación directa que realmente se produce entre los valores y, por ello, es preciso centrar el estudio en la matriz de valores de correlación. A continuación, en la Figura 3.25 se representa en forma de mapa de calor la misma y se puede determinar de forma clara que, en conjunto, la temperatura presenta una escasa correlación hacia la potencia generada, suponiendo un valor de 0,279. Para el resto de parámetros tampoco se mejora, lo que dificulta el análisis entre la producción energética y las condiciones climáticas si se emplean los datos de *SustDataED*.

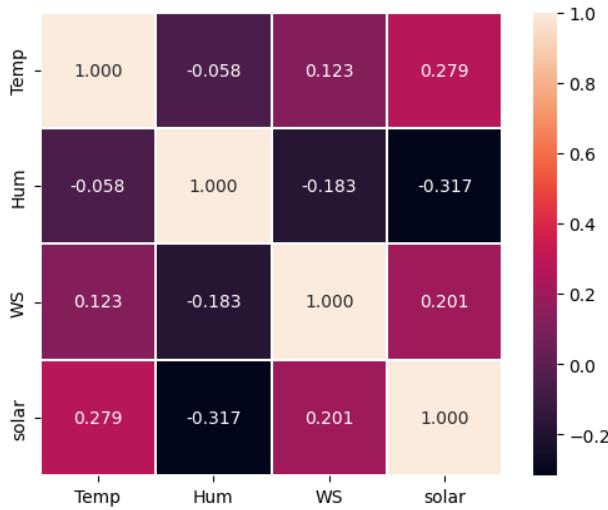


Figura 3.25: Representación de los valores de correlación mediante un mapa de calor (*SustDataED*)

3.3.2.2. Preprocesamiento de datos de *PVWatts*

El preprocesamiento de los datos extraídos de la herramienta *PVWatts* consisten principalmente en el formateo de los mismos y en la adecuación de los valores en cada instante para el posterior cálculo de las cargas. Como motivo de esto, se añaden las operaciones necesarias en el notebook *preprocessing_production.ipynb*.

Como se había expuesto en la Sección 3.2.2.3, además de en las Tablas 3.9 y 3.10, *PVWatts* proporcionaba dos datasets que replicaban un año completo de mediciones. En este caso, el preprocesamiento se va a enfocar en el que contiene las medidas adquiridas por hora para un año completo. Por tanto, teniendo en cuenta esta frecuencia de adquisición, no es necesario a priori sintetizar los datos como se había realizado en el caso de *SustDataED* (ver Sección 3.3.1.3), para el cual se transformaban muestras adquiridas cada minuto a valores promediados por hora.

Tampoco se requiere combinar los datos climáticos y ambientales con los valores de producción fotovoltaica, ya que el dataset los proporciona directamente todos en un mismo fichero (ver Tabla 3.10). No obstante, el paso que sí se debe realizar en esta fase de procesamiento es establecer una marca de tiempo específica a cada una de las filas de datos. Es decir, como la herramienta *PVWatts* genera un dataset a partir de la simulación de un sistema real de producción energética, las medidas dadas hacen referencia a una hora, día y mes, pero no se especifican para un año en concreto, sino que representan valores generalizados.

Considerando lo anterior, se debe añadir al dataset de origen una nueva columna dedicada a los valores de marca de tiempo de cada instante (*timestamp*) para poder indicar el año. Este paso es imprescindible para permitir posteriormente la comparación los datos de consumo con los de producción y establecer las cargas en el algoritmo DEN2NE, como se expondrá en las Secciones 3.3.3 y 3.5.1, respectivamente.

Es importante recordar en este punto que, para el desarrollo de este TFM, se ha seleccionado anteriormente el segundo despliegue de *SustDataED* y se ha reducido el manejo de dicho dataset a un rango temporal especificado entre los años 2010 y 2011. No obstante, en este paso del procesamiento del dataset de *PVWatts*, el fichero que se decide generar a la salida va a abarcar un conjunto de datos de producción con marcas de tiempo comprendidas entre los años 2010 y 2014.

En otros términos, se tienen en cuenta los cuatro despliegues para obtener el rango temporal completo de medidas que proporciona el dataset *SustDataED*. El motivo de reallizarlo de esta manera es prever la necesidad futura de emplear un despliegue diferente al seleccionado inicialmente, aportando así cierta flexibilidad. En la Figura 3.26 se representan los valores de potencia generada obtenidos en la simulación para el rango temporal del segundo despliegue.

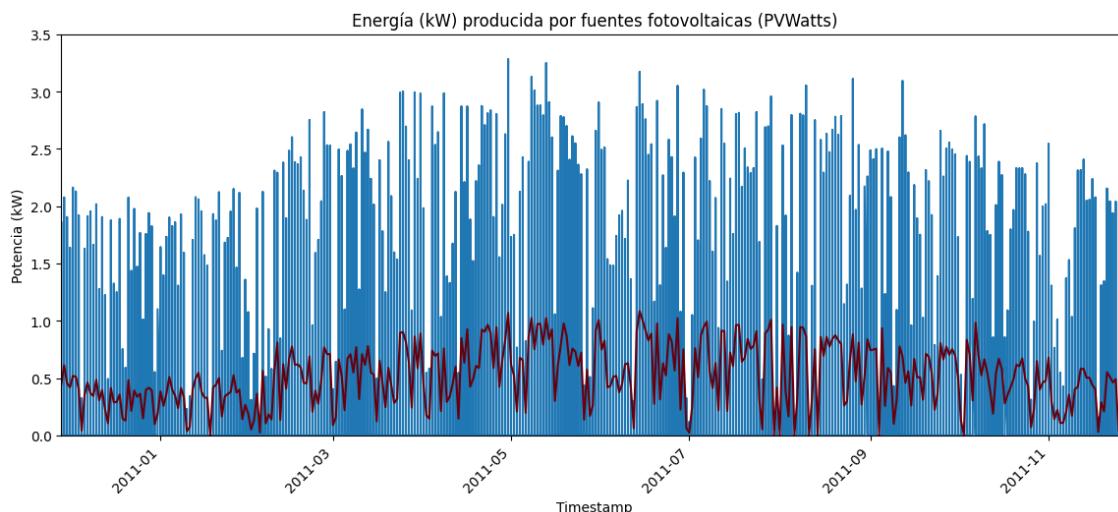


Figura 3.26: Gráfica de energía solar producida en el rango seleccionado (*PVWatts*)

Por otro lado, a modo de simplificar la información y de visualizar de una forma gráfica la correlación existente entre cada uno de los parámetros, se introduce la Figura 3.27 para representar la matriz de valores mediante un mapa de calor. Un valor igual a 1 indica

una correlación positiva perfecta y, como se puede apreciar, la radiación en cada instante incide de una manera directa en la cantidad de potencia que se genera. Coheramente, la más destacable a observar es la radiación en el plano del array POA, que alcanza un valor de 0,999. Se pueden observar estas correlaciones también, en las gráficas expuestas en la Figura 3.30.

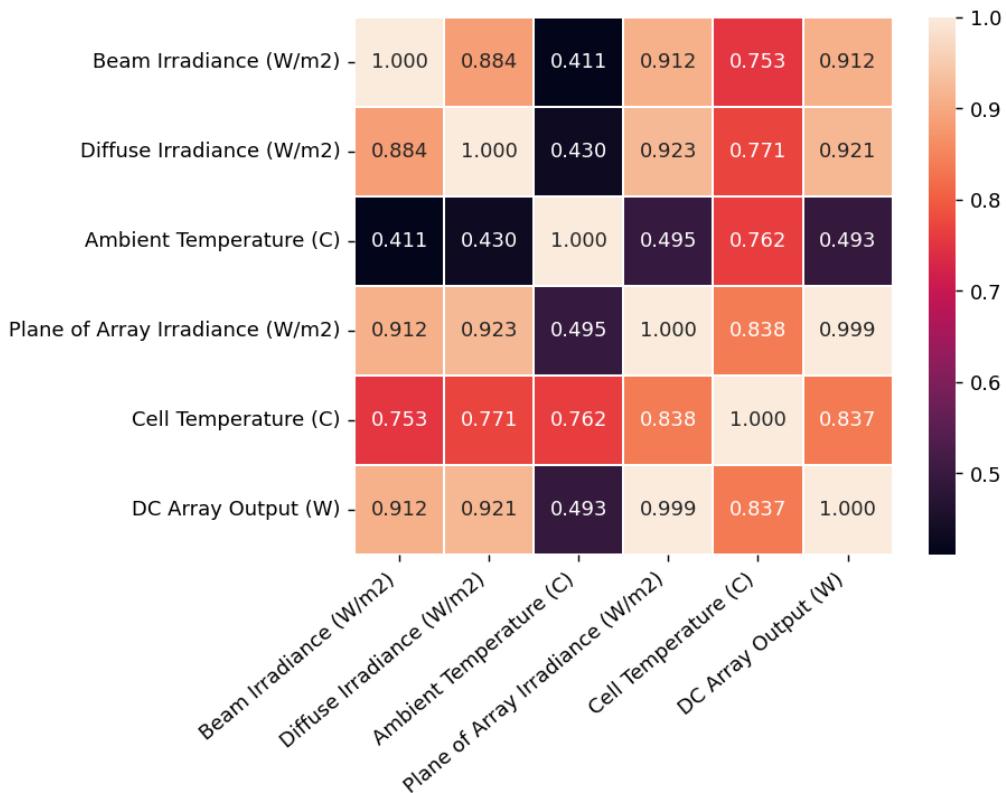


Figura 3.27: Representación de los valores de correlación mediante un mapa de calor (*PVWatts*)

De la misma forma que se ha realizado en la Sección 3.3.2.1 para los datos de *Sust-DataED*, se representa la primera semana de agosto a modo de ejemplo para observar la correlación entre los valores de potencia en cada instante y los parámetros ambientales y climáticos. En la Figura 3.28, se comparan los valores de temperatura, tanto ambiental como de la célula, con los de potencia generada en órdenes de kW. Se puede apreciar que, a mayores temperaturas adquiridas por la célula fotovoltaica, se incrementa la potencia generada. No obstante, es preciso indicar que la temperatura de las células puede llegar a ser un factor limitante si se supera el máximo recomendado, puesto que perjudicaría gravemente al rendimiento del sistema.

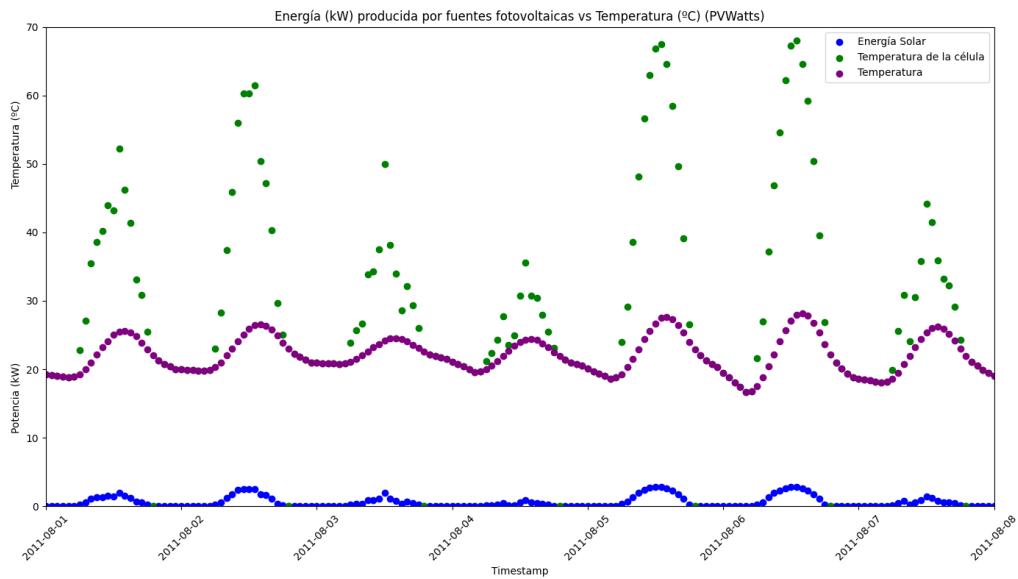


Figura 3.28: Correlación entre temperatura y energía solar producida en el mes de agosto (*PV-Watts*)

Por otro lado, en la Figura 3.29, se representan los valores de índice de radiación en el plano del array (POA) y de radiación difusa (DIF) frente a la potencia en cada instante. Como se expuso en la Sección 3.2.2.3, el primero está directamente relacionado con el índice de radiación directa normal (DNI) y se calcula como el producto del mismo por el ángulo de inclinación de las células. Debido a este motivo, y a que el DIF depende de la dispersión introducida por la atmósfera, se observa una mayor correlación entre los valores de energía generada y los de POA.

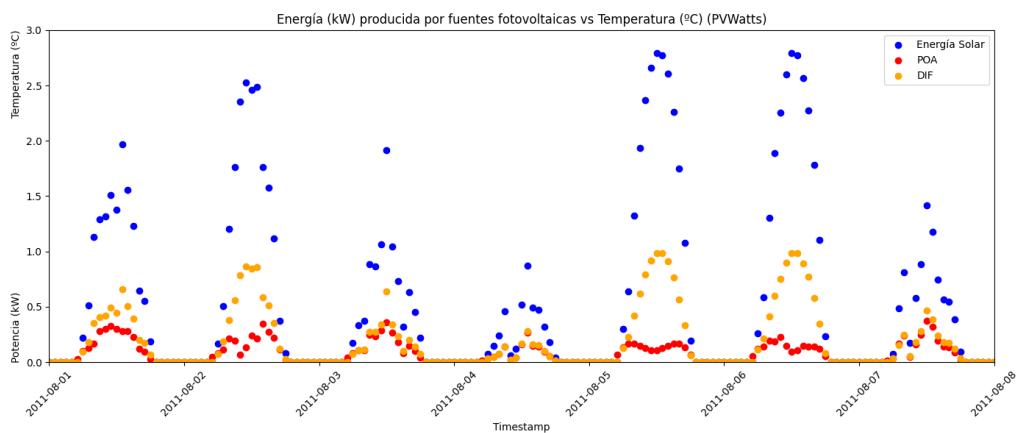


Figura 3.29: Correlación entre radiación y energía solar producida en el mes de agosto (*PVWatts*)

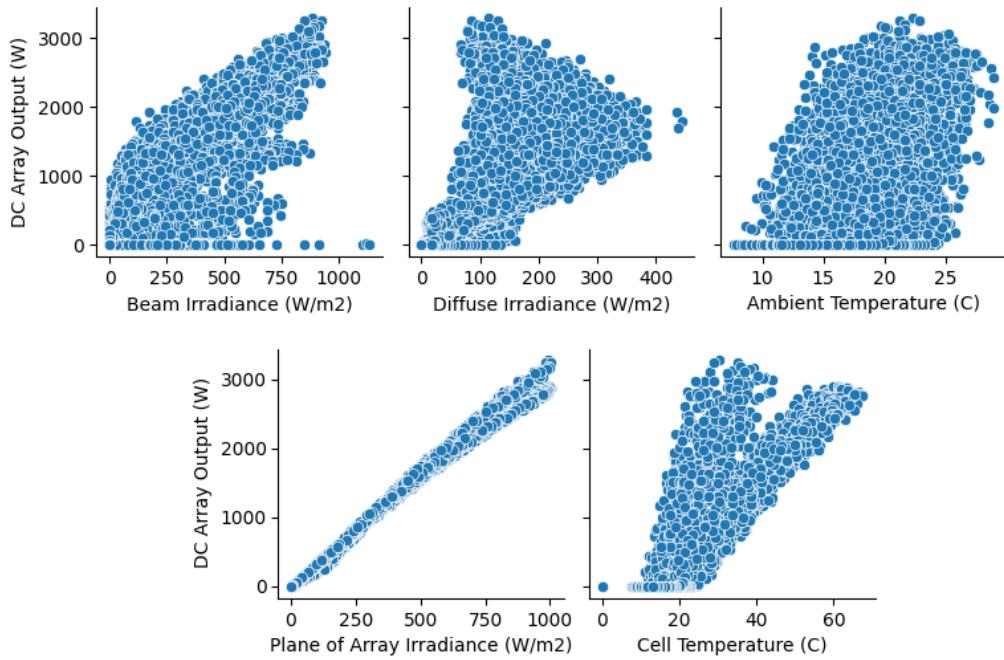


Figura 3.30: Representación de los valores de correlación mediante gráficas (*PVWatts*)

3.3.2.3. Selección de datos de producción

Como se había introducido en la Sección 3.3.2, se requería aplicar un preprocesamiento en relación con los datos de producción energética dados, tanto por el dataset *SustDataED*, como por la herramienta *PVWatts*, para poder llevar a cabo un análisis comparativo de ambos de una forma correcta. Una vez implementados estos pasos, se puede tomar la decisión de seleccionar la fuente de datos más adecuada para el desarrollo de este TFM, a partir de las características observadas y las representaciones gráficas expuestas en este documento. Añadiendo también a este análisis el estudio realizado en las Secciones 3.1.1 y 3.2.2, se determina finalmente, la elección de ***PVWatts*** como fuente de datos de producción energética.

Esta decisión se rige por la razón principal de la precisión que aportan los datos al tratarse de un dataset generado a partir de la simulación de un sistema real y de la configuración de múltiples parámetros de entrada, en referencia a la ubicación y a los requisitos técnicos deseados (ver Sección 3.2.2). Es decir, los resultados proporcionados por la herramienta han sido analizados en profundidad, además de contrastados con el estudio teórico, lo que justifica su precisión respecto a un caso real (ver Sección 3.2.2.3).

Otra ventaja que aporta el empleo de la herramienta *PVWatts* es la posibilidad de rediseño y de replicación de los datos anuales. En caso de desecharse, se permitiría simular otro tipo de sistema fotovoltaico real y determinar parámetros de entrada diferentes en la configuración, lo que supone un desarrollo mucho más flexible. Además, se aporta una mayor información que *SustDataED* sobre las condiciones climáticas que se dan en cada instante, sobre todo en términos de radiación. Esto es importante, ya que, como se podía apreciar en los mapas de calor expuestos (ver Figuras 3.25 y 3.27), los valores respectivos a los diferentes tipos de radiación presentan una gran correlación con la cantidad de potencia a la salida del sistema fotovoltaico. Así, se permite determinar de forma directa la potencia que se puede generar en cada instante. Sin embargo, no ocurría de la misma forma para los datos de temperatura dados por *SustDataED*, lo que suponía una gran limitación para analizar la información.

Por otro lado, el dataset *SustDataED*, a diferencia de *PVWatts*, sí que se basa en medidas de un despliegue de viviendas real, pero los valores de generación energética que aporta se definen en términos globales a todas las hogares. No se aporta información exacta sobre el número de viviendas a los que se abastece, ni sobre las dimensiones del sistema o sistemas fotovoltaicos de los que se han adquirido los datos en cada instante. En consecuencia, no se puede evaluar su utilidad específicamente en un entorno de SGs como motivo de los objetivos de este TFM y se introducen dificultades para analizar los valores de potencia dados. Por todo esto, se decide reducir el uso del dataset *SustDataED* a los datos de consumo de los hogares.

3.3.3. Combinación de datos de consumo y de producción

Se introduce la programación de un nuevo *notebook*, denominado como *power_selection*, que va a determinar los ficheros con los valores de potencia finales mediante la combinación de los datos de consumo y de producción preprocesados. Es decir, en este punto se parte, por un lado, de los 23 ficheros de consumo (*consum_x.csv*), obtenidos de *SustDataED* y que han sido creados para representar el comportamiento eléctrico de las 23 viviendas que componen el segundo despliegue. Por otro lado, se dispone del dataset final de producción, extraído de la herramienta *PVWatts* y preprocesado para ser aplicado al rango temporal seleccionado.

Por tanto, el objetivo del *notebook* en cuestión se basa en obtener a la salida 23 nuevos ficheros que contengan los valores de cargas netas para cada instante, las cuales se establecerán posteriormente en el algoritmo DEN2NE (ver Sección 3.5.1). Para ello, es necesario combinar cada fichero de consumo con los datos de producción y, después, aplicar la diferencia entre la potencia generada y consumida a partir de la marca de tiempo dada en

cada fila. Se escoge como ejemplo la vivienda con identificador 1 para representar en las Figuras 3.31 y 3.32 los valores de carga resultante a partir de los ficheros de entrada. En el caso de la Figura 3.32, se muestran también de forma continua los valores promedios diarios de carga.

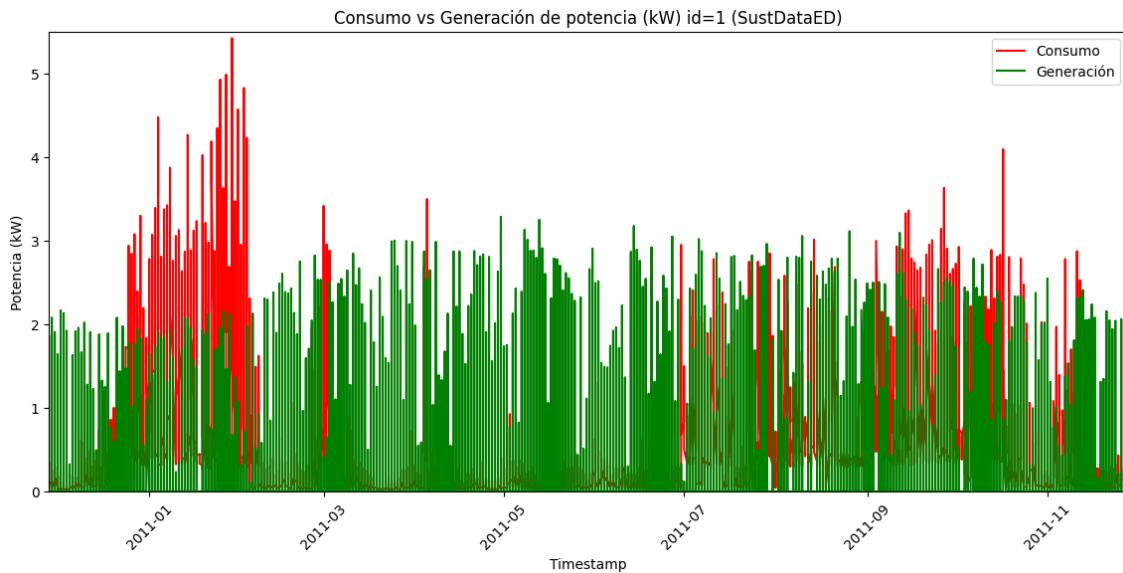


Figura 3.31: Comparación entre el consumo y la generación de potencia en la vivienda 1 en el rango seleccionado

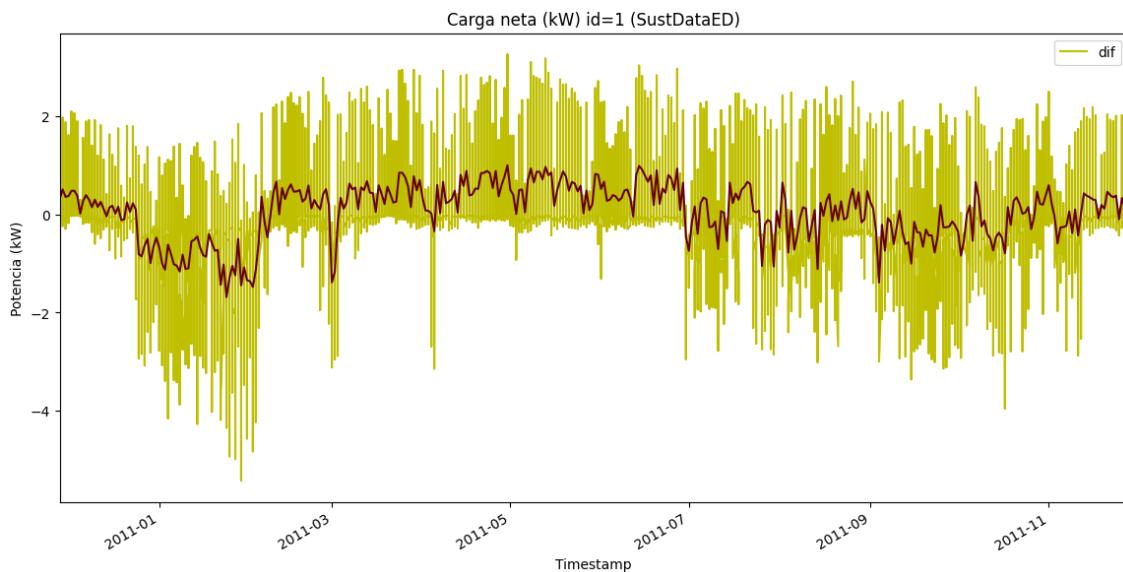


Figura 3.32: Gráfica de la carga neta resultante en la vivienda 1 en el rango seleccionado

Como se realiza una operación de combinación de los datos de consumo y de producción, los campos de los nuevos ficheros de cargas suponen los definidos, tanto en la Tabla 3.14, como en la Tabla 3.10, exceptuando algunos parámetros que se descartan para el desarrollo, como son el albedo o la velocidad del viento. En la Tabla 3.16, se especifican los campos definitivos.

Campo	Descripción	Unidades
<i>timestamp</i>	Instante temporal de medida	datetime
<i>datetime</i>	Fecha del valor promedio	datetime
<i>H</i>	Hora del valor promedio	-
<i>iid</i>	Identificador de vivienda	-
<i>Diffuse Irradiance</i>	Índice de radiación difusa (DIF)	W/m ²
<i>Plane of Array Irradiance</i>	Índice de radiación en el plano del array (POA)	W/m ²
<i>Ambient Temperature</i>	Temperatura ambiente	C
<i>Cell Temperature</i>	Temperatura de las células solares	C
<i>DC Array Output</i>	Potencia de salida DC del array	W
<i>AC System Output</i>	Potencia de salida AC del sistema	W
<i>Pavg</i>	Potencia consumida	W
<i>Dif</i>	Carga neta calculada	W

Tabla 3.16: Dataset resultante de la etapa de procesamiento

Por otro lado, en cuanto a la nomenclatura, cada fichero se nombra como *load_x.csv*, donde x hace referencia al identificador de la vivienda a la que pertenecen las cargas calculadas. Por tanto, se puede expresar que, después de toda la fase de procesado, se consigue el objetivo de obtener un dataset final compuesto por 23 ficheros con cargas, el cual supondrá la base principal de la operativa del algoritmo DEN2NE para construir el conjunto de datos a entrenar.

Adicionalmente, a modo de simplificar la compresión de toda la secuencia de acciones que se han llevado a cabo durante la Sección 3.3, se representa en la Figura 3.33 el diagrama completo. En el mismo, se detallan los datos de origen empleados, los *notebooks* de *Jupyter* creados y ejecutados en cada paso de procesamiento y los resultados en forma de nuevos ficheros .csv que derivan hasta obtener finalmente los 23 ficheros *load_x.csv*⁵.

⁵<https://github.com/PaulaBartolomeMora/TFM/tree/main/SustData/dataset/nodes/loads>

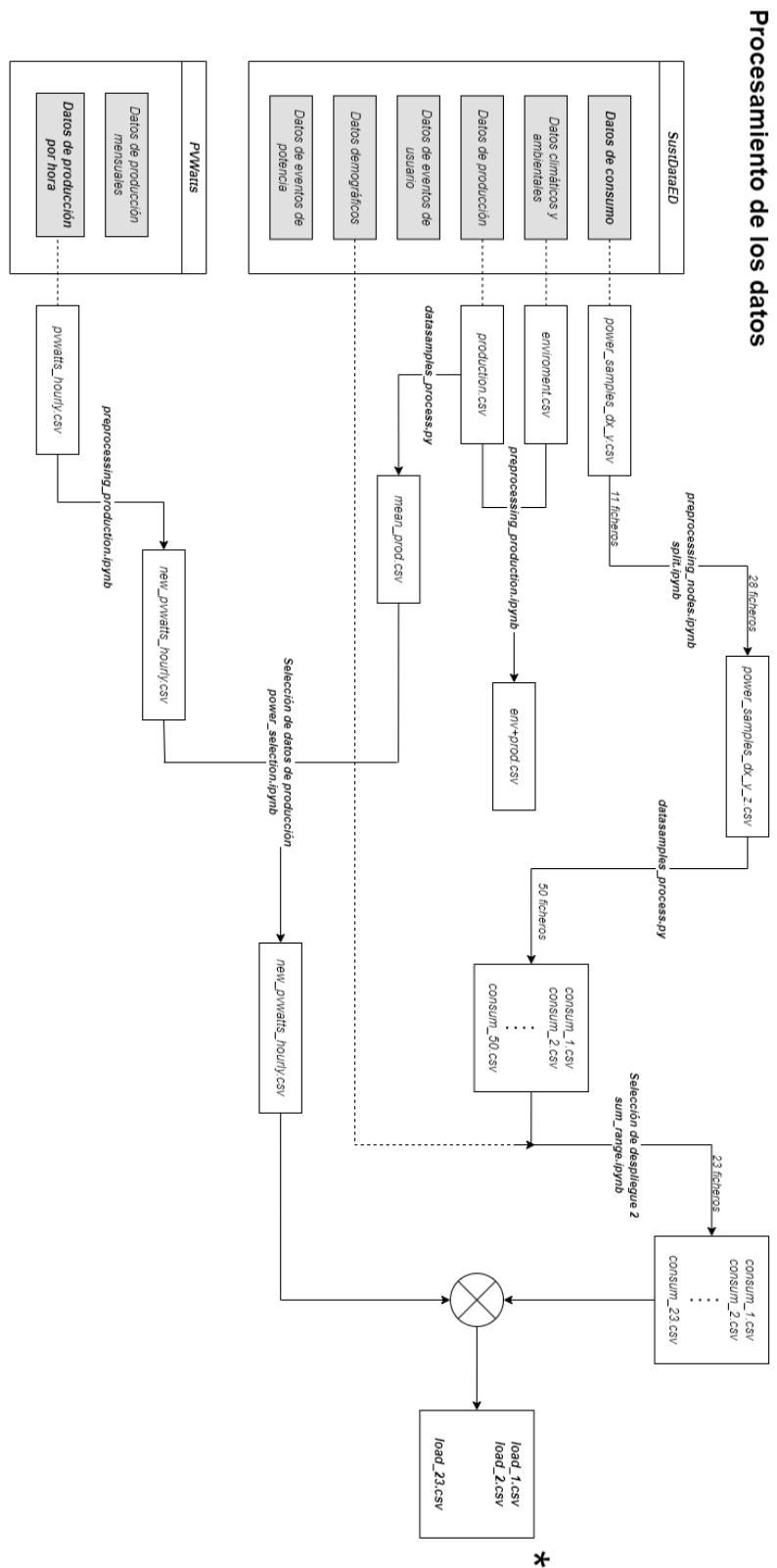


Figura 3.33: Diagrama completo de diseño y procesamiento para la obtención de los ficheros *load_x.csv*

3.4. Planteamiento de escenarios y generación de topologías en BRITE

La presente Sección está dedicada al planteamiento de una serie de escenarios de red y a la consecuente generación de topologías mediante el empleo de la herramienta BRITE, cuyo funcionamiento ha sido descrito en la Sección 2.6.1. La importancia de esta fase del diseño viene dada por la necesidad de aplicar la operativa del algoritmo DEN2NE sobre múltiples topologías para obtener, finalmente, el dataset sobre el que se entrenarán y desarrollarán los modelos de ML y DL. Se debe tener en cuenta que, para que este conjunto de datos final permita aportar suficiente información útil a los modelos, el número de topologías aleatorias a generar en BRITE debe de ser relativamente elevado.

3.4.1. Planteamiento de escenarios y configuración de BRITE

Para emplear la herramienta, se deben plantear los escenarios de red sobre los que se desea basar la generación de topologías aleatorias. Por ello, es preciso definir la configuración de los parámetros de entrada en el script *autogenerador.sh*, tomando en consideración el requerimiento anterior. Como se especificaba en la Sección 2.6.1.3, este script está dedicado a la automatización de la ejecución, tanto de la herramienta, como del *parser*, para obtener a la salida los ficheros finales con las posiciones de los nodos en el plano y con la información relativa a las distancias y a los nodos que se interconectan con cada enlace (*Nodos.txt* y *Enlaces.txt*).

Entonces, se puede expresar que el cálculo del número total de topologías que se pueden generar con BRITE resultará de operar el producto de los siguientes parámetros de entrada configurados:

- Número de modelos de topología a emplear: Como se había introducido en la Sección 2.6.1.2, este TFM se va a basar en el empleo de topologías aleatorias a nivel de router y, en particular, en los modelos Router Waxman y Router Barabasi-Albert.
- Número de dimensiones: Para las pruebas a simular en el algoritmo DEN2NE, se pretende manejar topologías con una gran cantidad de nodos, pero reduciendo los saltos de incremento del total. Por ello, se configura una generación desde 100 a 200 nodos con un incremento de 50 en 50, suponiendo la configuración de 3 dimensiones de topologías (100, 150 y 200 nodos).
- Número de grados de conectividad: Este parámetro determina el número de enlaces por nodo y se especifican 3 grados m diferentes en el script. Es preciso indicar

que, como en DEN2NE se tratarán los enlaces como bidireccionales, realmente cada topología tendrá un grado $2m$.

- Número de semillas de generación: Es importante destacar que para la generación de topologías aleatorias se aplican 10 ficheros de semillas. Esto tiene como resultado la generación de 10 topologías diferentes para cada uno de los escenarios configurados o, en otros términos, para cada una de las configuraciones posibles de los parámetros de entrada.

Adicionalmente, se deben determinar otros parámetros que hacen referencia a los nodos, como son el modo de introducción al plano y su posicionamiento. Respectivamente, se establece una introducción incremental y un posicionamiento totalmente aleatorio alrededor del plano. También, los parámetros específicos del modelo Router Waxman, α y β , que toman valores de 0,2 y 0,15, respectivamente.

Código 3.10: Configuración de los parámetros de entrada en el script de automatización de BRITE

```

1 topologia_rt_waxman=1 # Empleo de modelo RTWaxman
2 topologia_rt_barabasi=2 # Empleo de modelo RTBarabasi
3 nodos=$((seq 100 50 200)) # Número de nodos por topología (dimensiones)
4 m=(1 2 3) # El grado de conectividad real es 2, 4, 6
5 n_topologias_distintasxnode=10 # Número de topologías aleatorias en función del número de ↵
    ↵ semillas
6
7 NodePlacement=1 # Posicionamiento aleatorio de los nodos
8 GrowthType=1 # Modo de introducción incremental
9
10 #parámetros específicos de Waxman
11 alpha=0.2
12 beta=0.15

```

3.4.2. Resultados de la generación de topologías aleatorias

Considerando los parámetros de entrada configurados anteriormente, se puede cuantificar el número total de topologías que resultará de la ejecución del script *autogenerador.sh* a partir de la siguiente expresión:

$$N_{topos} = N_{modelos} \times N_{dimensiones} \times N_{grados} \times N_{seeds_gen}$$

$$N_{topos} = 2 \times 3 \times 3 \times 10 = 180$$

Es preciso indicar que en el caso de requerirse un mayor número de topologías a simular en DEN2NE, bastaría únicamente con modificar el rango establecido para el número de

nodos o el valor del incremento de los mismos. No obstante, puesto que se pretende tener en cuenta para la ejecución del algoritmo el dataset resultante de la etapa de procesamiento (ver Sección 3.3.3), no se precisa aumentar el número de topologías aleatorias a priori. Estos motivos vendrán justificados detalladamente en la Sección 3.5.2, dedicada a la configuración de los parámetros de entrada de DEN2NE.

De forma adicional, para observar gráficamente los resultados que se obtienen a la salida se representan varios ejemplos de topologías generadas en la Figura 3.34. Respectivamente, las gráficas obtenidas de *matlab* hacen referencia al modelo Barabasi-Albert y al Waxman y presentan una configuración de 100 nodos y un grado de conectividad $m=2$, que al suponer enlaces direccionales realmente toma el valor de 4. Es importante indicar que, para apreciar las diferencias existentes entre ambos modelos, se selecciona el mismo fichero de semilla (*seed_4*) para la representación.

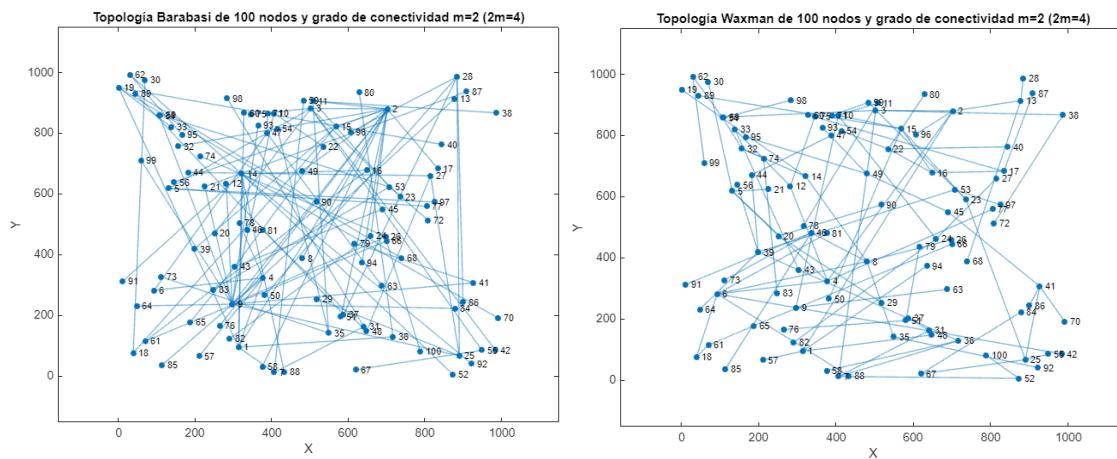


Figura 3.34: Representación de diferencias de los modelos Router Barabasi-Albert y Waxman a partir de los mismos parámetros de entrada

3.5. Simulación de las topologías en DEN2NE

Esta Sección viene dada por el objetivo de obtener el conjunto de datos final sobre el que se basará el desarrollo en el Capítulo 4. Para lograrlo, se requiere llevar a cabo un gran número de simulaciones en DEN2NE a partir de las topologías generadas con la herramienta BRITE (ver Sección 3.4). No obstante, previamente a la ejecución del algoritmo, es imprescindible implementar una serie de modificaciones en el mismo para ajustar su funcionamiento para obtener resultados de utilidad para el presente TFM. La

secuencia de acciones que se llevará a cabo se detallará en la Sección 3.5.1. De la misma forma, se justificará el criterio escogido para determinar cuándo se producen errores en el proceso de distribución energética. Esto servirá de base para después entrenar los modelos de ML y DL.

Por consiguiente, se añade la Sección 3.5.2, donde se especificará la configuración de los parámetros de entrada de DEN2NE y se cuantificará el número total de pruebas posibles que se podrían llegar a realizar a partir de las topologías generadas y de los datos reales procesados anteriormente. Se comprobará a través de diferentes pruebas que las modificaciones introducidas en el algoritmo DEN2NE producen una operativa acorde a las necesidades de este TFM. De forma concluyente, a partir de los resultados obtenidos en las simulaciones, se añadirá la Sección 3.5.3, en referencia a las características del dataset final que se utilizará para el entrenamiento de los modelos.

3.5.1. Adaptación del algoritmo DEN2NE a las pruebas

Todas las modificaciones realizadas en el funcionamiento del algoritmo DEN2NE han sido aplicadas a partir de los ficheros contenidos en el repositorio⁶ del equipo de investigación NetIS de la UAH y recopiladas en el directorio de ficheros fuente del repositorio⁷ dedicado a este TFM.

3.5.1.1. Importación de los perfiles de carga reales

La primera modificación que se debe implementar en el funcionamiento de DEN2NE consiste en la importación de los valores de carga reales, obtenidos como resultado de llevar a cabo la fase de procesamiento de los datos (ver Sección 3.3.3). La necesidad de aplicar este paso se debe a que el algoritmo, a partir de una topología dada a la entrada, establece una función de densidad de probabilidad para determinar de forma aleatoria un valor de carga para cada uno de los nodos de una topología. Para modificar este proceso se sigue la siguiente secuencia de pasos:

1. Definición de función *getLoads_Config* en *dataCollector.py*: Se añade una función de recolección de las configuraciones de carga reales, pasando como argumentos el directorio definido en *path_simtests* y el fichero de pruebas *sim_file*. Después, se inicializa una variable de diccionario y se almacenan los valores de potencia generada, consumida y la diferencia calculada de ambas. Estos valores se manejan en términos de kW, puesto que DEN2NE está configurado para trabajar en esta unidad.

⁶<https://github.com/NETSERV-UAH/den2ne-Alg>

⁷<https://github.com/PaulaBartolomeMora/TFM/tree/main/den2ne/src>

2. Definición de funciones *cargas* y *cargas_con_límite* en *brite_intf.py*: Ambas funciones se encargan de aplicar los valores de carga reales, extraídos de la función anterior, a cada uno de los nodos de una topología determinada. En el caso de las simulaciones que se realizan en este TFM, se hará uso de la primera, debido a que no se configuran límites de los valores de carga (ver Sección 3.5.2). Por tanto, poniendo el foco en la función *cargas*, se pasa como argumento el diccionario resultante de la función *getLoads_Config* y se extrae a partir de las claves del mismo el número de perfiles de carga (23) a manejar. De la misma forma, se pasa también como argumento el fichero de nodos, anteriormente generado con BRITE y el *parser*, para conocer las dimensiones que tiene la topología en cuestión. Con este dato se determina la cantidad de iteraciones que son necesarias para aplicar el valor de carga real a todos los nodos de una topología y, para cada uno de ellos, se determina de forma aleatoria uno de los 23 perfiles reales de carga. A la salida, se obtiene un nuevo diccionario con tamaño igual al número de nodos que hay en la topología y que viene constituido por la información del identificador del perfil real seleccionado y del valor de carga neta aplicada.
3. Definición de la variable *id_orig* en *node.py*: Se añade al constructor de la clase de nodo que tiene DEN2NE un nuevo elemento, en relación al identificador del perfil de carga seleccionado para un nodo en cuestión.
4. Introducción de la variable *id_orig* en la función *buildGraph* en *graph.py*: La nueva variable de nodo se incluye en el proceso de generación del grafo para mantener el conocimiento del perfil de carga que se ha seleccionado para cada nodo durante la ejecución completa del algoritmo.
5. Modificaciones en *test_topo.py*: El fichero dedicado a las pruebas de DEN2NE debe incluir las llamadas a las funciones creadas o modificadas anteriormente. También, requiere definir los nuevos parámetros *path_simtests* y *sim_file*, en referencia a los ficheros de prueba que contienen los valores de carga, extraídos del dataset para un instante temporal determinado (ver Sección 3.5.3).

3.5.1.2. Introducción de la etiqueta de error

Como se había introducido en el Capítulo 1, el objetivo de este TFM viene dado por la necesidad de poder detectar y predecir los errores que se pueden producir durante el proceso de distribución energética que realiza DEN2NE. Por ello, una vez que se aplican los pasos anteriores, se debe definir el criterio de error a partir del cual se van a etiquetar los resultados extraídos del algoritmo. Teniendo en cuenta que para las simulaciones se configura un escenario real con pérdidas y limitación de capacidad de los enlaces (ver

Sección 3.5.2), se toma la decisión de establecer la condición de fallo en base a la existencia de un exceso de capacidad en un intercambio energético entre dos nodos:

1. Introducción de la variable *link_overflow* en la función *globalBalance* en *den2neALG.py*: Para cada intercambio energético que se realiza en el proceso de distribución, se comprueba que la carga direccionada desde el nodo de origen al nodo destino no sobrepasa la capacidad del enlace que los interconecta. En caso afirmativo, se activa la variable *link_overflow* y, por el contrario, se deja con valor nulo, indicando que el intercambio se ha producido sin fallos.
2. Modificación de la función *getLosses* en *link.py*: DEN2NE, en su funcionamiento original, calcula las pérdidas del enlace a partir del valor de carga intercambiado o de la propia capacidad, en función de sobrepasarla o no. A modo de depuración y de simplificar su funcionamiento, se incluye como argumento la etiqueta *link_overflow* y se modifica la estructura de la función.

3.5.1.3. Extracción de resultados

Para extraer de la ejecución del algoritmo unos resultados que contengan toda la información útil sobre la que se pueda crear el dataset final, es preciso aplicar las siguientes modificaciones:

1. Definición de la función *getLinkDist* en *graph.py*: Se incluye una función para obtener la distancia existente entre el nodo de origen y de destino.
2. Introducción de la variable *data_topo* en la función *globalBalance* en *den2neALG.py*: Se define el diccionario *data_topo* y, para cada intercambio energético, se almacenan en el mismo la información del nodo origen y destino (etiquetas jerárquicas y longitudes de las mismas), los datos del enlace (distancia y capacidad), el valor de la etiqueta de error y la carga que se intercambia. En el caso de las longitudes de las etiquetas jerárquicas, se añaden las líneas de código necesarias en *globalBalance* para extraer las mismas.
3. Modificaciones en *test_topo.py*: Se cambia la llamada a la función *globalBalance* para poder extraer el diccionario *data_topo* de la misma. También, se establece la nomenclatura que tendrán los ficheros de resultados. Como se expondrá más adelante en la Sección 3.5.3, esta nomenclatura se determina en base a la información dada por el instante temporal del fichero de test y por el resto de parámetros configurados en el script de automatización de pruebas *auto_run.sh*.

-
4. Modificaciones en *auto_run.sh*: Se añaden los parámetros que hacen referencia al directorio donde se ubican los ficheros de test y a cada uno de los mismos para automatizar las pruebas (ver Sección 3.5.3).

3.5.1.4. Configuración de la capacidad de los enlaces

Como paso adicional, se modifican las capacidades de los enlaces que se proporcionan por defecto en el fichero *links_config.csv*. Esto se realiza con el objetivo de que el algoritmo configure enlaces de menor capacidad con la misma probabilidad que los de mayor, puesto que este proceso es aleatorio. A modo de simplificación, se establecen los tres tipos de enlaces definidos en la Tabla 3.17

<i>R</i> (ohm/km)	<i>I max</i> (A)	Sección (mm ²)
0,272	185	70
0,78	100	25
1,91	53	10

Tabla 3.17: Configuraciones de enlaces de *links_config.csv*

3.5.2. Configuración de los parámetros de entrada

Para ejecutar DEN2NE, es preciso definir la configuración de los parámetros de entrada en el script *auto_run.sh*. Este script se diseña con el objetivo de automatizar las simulaciones en el algoritmo y, como se ha expuesto en la Sección 3.5.1, ha sido necesario ajustar el mismo a los requerimientos de las pruebas de este TFM. El cálculo de la cantidad total de simulaciones que se pueden ejecutar en DEN2NE viene dado por el producto de todos los parámetros siguientes:

- El número total de topologías generadas: En la Sección 3.4.2, se especifica una cantidad total de 180 topologías obtenidas a la salida de la herramienta BRITE.
- El número de instantes temporales del dataset: El conjunto de datos resultante de la etapa de procesamiento, detallado en la Sección 3.3.3 y, específicamente, en la Tabla 3.16, abarca una cantidad de filas igual al total de instantes temporales que se han tomado en consideración. En otros términos, al tratarse de muestras tomadas cada hora durante un rango temporal que comprende un año completo, su valor se calcula como 24x365=8760 instantes.
- El número de criterios: Como se detallaba en la Sección 2.3, DEN2NE presenta 6 criterios de selección de los mejores caminos hacia el nodo raíz. En este caso, se

implementan a las pruebas los 6 tipos.

- El número de tipos de escenarios de red: En la Sección 2.3, también se exponían los 4 tipos de escenarios que permite configurar el algoritmo. Teniendo en cuenta que las simulaciones deben acercarse a un entorno de SG real y que el objetivo se basa en encontrar las transacciones energéticas entre nodos que superen la capacidad del enlace, se determina únicamente el escenario de pérdidas y límite de capacidad.
- Modo de limitación de carga: Se especifica únicamente el modo que determina valores de carga ilimitados.
- El número de semillas de ejecución: De la misma manera que se ha especificado anteriormente para BRITE (ver Sección 3.4.1), se aplica al algoritmo DEN2NE un conjunto de archivos de semillas para obtener simulaciones diferentes a partir de una misma topología a la entrada. En este caso, debido a la cantidad de datos que ya se maneja y, para no introducir latencias considerables en el lanzamiento de las pruebas, se especifican 5 semillas por topología.

En el caso de los parámetros de entrada referentes a las topologías, como son el número de nodos por topología, los modelos a simular y el número de semillas de generación, se sigue la misma configuración especificada en la ejecución de BRITE (ver Sección 3.4.1). No obstante, para el grado de conectividad se determinan directamente los valores reales, suponiendo enlaces bidireccionales.

Código 3.11: Configuración de los parámetros de entrada en el script de automatización de DEN2NE

```

1 TOPO_CRITERIONS=(0 1 2 3 4 5) # Criterios de selección de IDs
2 TOPO_BEHAVIORAL=3 # Tipo de escenario de red = modo Losses and Capacity (3)
3 TOPO_LOAD_LIMIT=0 # Sin límite de carga
4 TOPO_RUNS=5 # Seeds de ejecución de DEN2NE
5
6 TOPO_NAMES=('barabasi' 'waxman') # Empleo de modelos RTWaxman y RTBarabasi (igual que BRITE)
7 TOPO_NUM_NODES=$((seq 100 50 200)) # Número de nodos por topología (igual que BRITE)
8 TOPO_DEGREES=(2 4 6) # El grado de conectividad real (en BRITE 1, 2, 3)
9 TOPO_SEEDS=(1 2 3 4 5 6 7 8 9 10) # Seeds de generación de BRITE

```

Por tanto, teniendo en cuenta la configuración implementada para los parámetros de entrada, se puede determinar el número total de simulaciones únicas posibles a partir de la siguiente expresión:

$$N_{sim} = N_{topos} \times N_{instantes} \times N_{criterios} \times N_{escenarios} \times N_{limit} \times N_{sem_ejec}$$

$$N_{sim} = 180 \times 8760 \times 6 \times 1 \times 1 \times 5 = 47.304.000$$

3.5.3. Creación del dataset final y conclusiones de las pruebas

Tomando en consideración el número total de simulaciones únicas posibles calculado en la Sección 3.5.2, se debe decidir qué pruebas se van a realizar en el algoritmo para construir el dataset final, puesto que es ineficiente e inviable ejecutar todas las posibilidades. Por lo tanto, se opta por escoger 12 instantes temporales, haciendo referencia a una hora determinada de un día por cada mes.

Por simplificar, como el rango temporal de las muestras (ver Sección 3.3.1.5) comienza el 28 de noviembre de 2010, se decide seleccionar los días 28 de cada mes. De la misma forma, para la hora se escogen las 11:00, ya que este es el momento del día en el que se aprecia una mayor producción energética de media (ver Figura 3.10). En consecuencia, en los resultados de la ejecución de DEN2NE se obtendrá un mayor número de intercambios energéticos con fallos, al existir una mayor probabilidad de exceder la capacidad de los enlaces. Esta selección es importante para conseguir un dataset final sobre el que se pueda entrenar de una forma correcta los modelos del Capítulo 4. No obstante, previamente a detallar la secuencia de acciones realizadas para construir este conjunto de datos final, se debe introducir el diagrama de la Figura 3.35 para aportar una mayor comprensión de esta Sección.

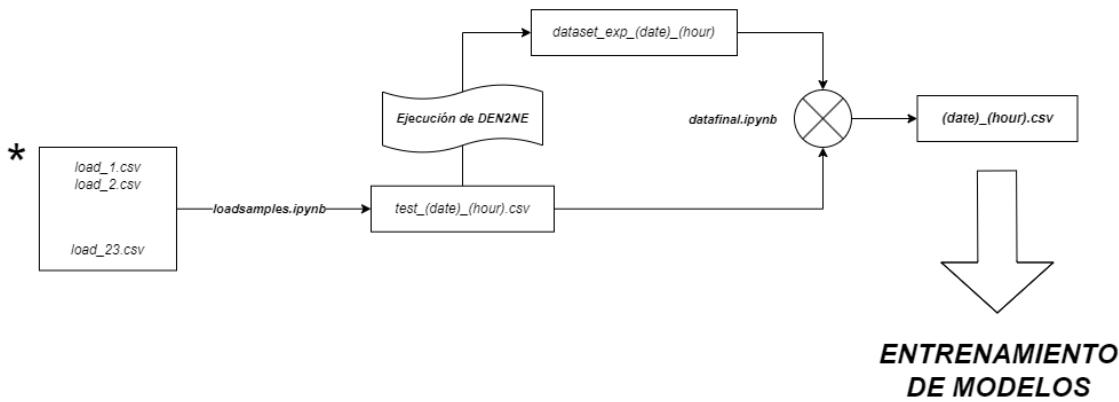


Figura 3.35: Diagrama completo de diseño del dataset final

Como el objetivo se basa en extraer de los ficheros finales de cargas netas `load_x.csv` (ver Sección 3.3.3) los datos de los instantes temporales especificados, se introduce un nuevo *notebook*, denominado como `loadsamples.ipynb` en el repositorio⁸. A partir del mismo, se

⁸<https://github.com/PaulaBartolomeMora/TFM/tree/main/SustData>

filtran los 12 instantes seleccionados y se obtienen a la salida 12 ficheros de test (ej. *test_2010-11-28_11.csv*), los cuales serán proporcionados a la entrada del algoritmo para importar los 23 perfiles reales de carga para cada prueba. Es en este paso cuando se proceden a ejecutar las simulaciones en DEN2NE.

La aplicación de cada fichero de test a la entrada proporciona a la salida un nuevo directorio nombrado con el instante temporal que especifica el fichero de test en cuestión (ej. *dataset_exp_2010-11-28_11*). En su interior, se impone una organización de los resultados en 18 carpetas, en función del modelo y del grado de conectividad al que hacen referencia (ej. *barabasi-200-6*). Cada una de dichas carpetas consta de 60 ficheros de resultados, que incluyen en su nombre el número de semilla, el criterio del algoritmo, el instante temporal, el grado de conectividad y el modelo (ej. *dataset_seed_1_cr_0_t_2010-11-28_11_dg_2_m_barabasi.csv*). En la Figura 3.36, se expone la nomenclatura especificada para cada uno de los directorios y ficheros de resultados recopilados en el directorio de resultados del repositorio⁹.

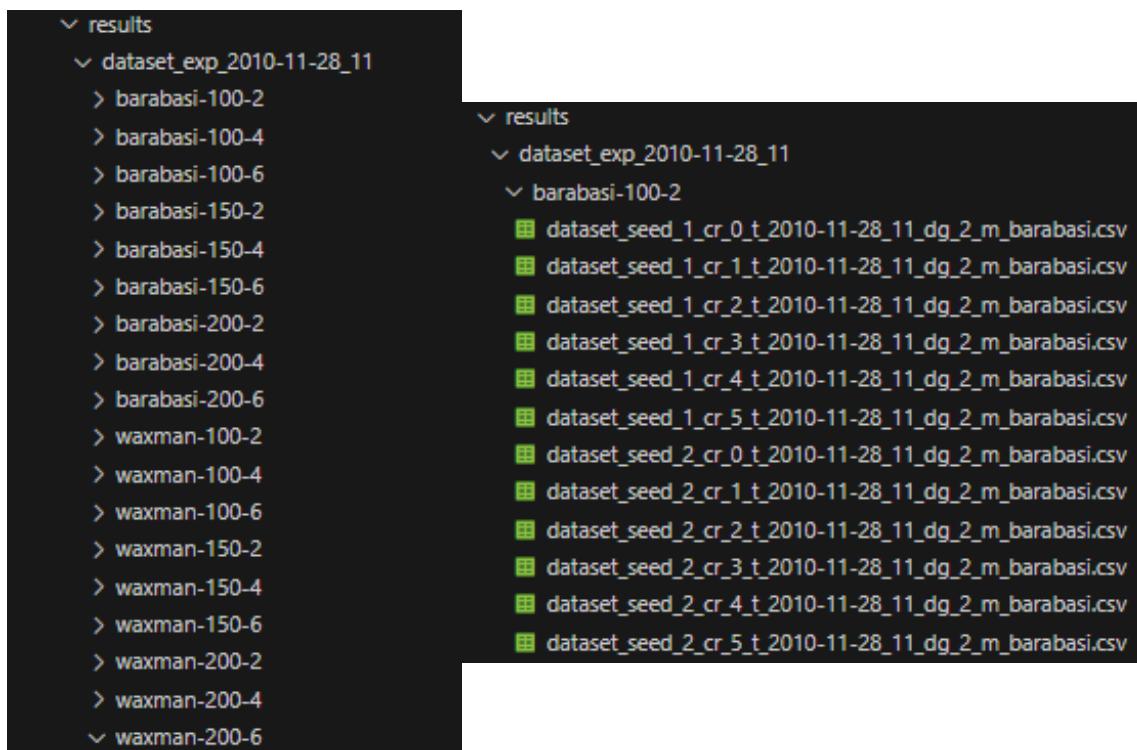


Figura 3.36: Nomenclatura de los directorios y ficheros de resultados

⁹<https://github.com/PaulaBartolomeMora/TFM/tree/main/den2ne/src/results>

Entonces, se puede expresar que para cada prueba ejecutada o fichero de test a la entrada de DEN2NE, se obtienen a la salida un total de 1080 ficheros de resultados. Cada uno de ellos presenta un número de filas igual a la cantidad de nodos de la topología simulada, o expresado de otra forma, igual al total de intercambios energéticos realizados (100, 150 o 200 filas). Con esto, ya es posible crear el dataset final mediante un nuevo *notebook*, denominado como *datafinal.ipynb*.

A modo de simplificar el tratamiento de los resultados del algoritmo, se agrupa primero, en un mismo *dataframe* el contenido de los 1080 ficheros resultantes de cada prueba. Despues, se comprueba que el porcentaje total de intercambios erróneos o con exceso de capacidad de enlace no es nulo para la prueba en cuestión. Es decir, si no se obtiene ninguna fila con la etiqueta *link_overflow* activa, no se podrán predecir errores cuando se desarrollen y se entrenen los modelos y, por ello, se debería utilizar otro instante temporal para las simulaciones.

Una vez realizada esta comprobación, la siguiente acción se basa en combinar los ficheros de resultados con el resto de medidas que proporcionan los ficheros de test. Las filas de ambos se agrupan a partir del perfil de carga configurado en el nodo origen, especificado por el valor del identificador real. Tras esta operación, el nuevo *dataframe* en *datafinal.ipynb* se constituye por todas las columnas de datos. En este paso, es importante revisar y eliminar aquellas que están replicadas, como ocurre en el caso del identificador y la fecha.

Código 3.12: Combinación de ficheros de resultados y de test

```
1 df_merged = pd.merge(df, df_sim, left_on='origen_id', right_on='iid', how='outer')
```

Finalmente, se almacena el *dataframe* ya limpio en un fichero, cuyo nombre hace referencia al instante temporal de la prueba en cuestión (ej. *2010-12-28_11.csv*). Este fichero contiene un total de 160.920 filas con un valor de etiqueta de error y con toda la información respectiva a cada intercambio energético simulado. Por ello, se define como el conjunto de datos final resultante de una prueba o instante temporal determinado. Como se han ejecutado 12 pruebas, si se agrupan los 12 conjuntos finales, se puede calcular una cantidad total de 1.931.040 intercambios etiquetados sobre los que entrenar los modelos. De forma adicional, se añade la Tabla 3.18 para exponer los campos que contienen cada uno de los conjuntos finales.

Con la obtención de estos 12 ficheros finaliza la etapa de diseño y, por ende, el presente Capítulo. A partir de los resultados obtenidos, se puede expresar de forma concluyente que se cumple el objetivo principal de generar un dataset completo, sobre el cual se entrenarán y desarrollarán los modelos en el Capítulo 4.

Campo	Descripción	Unidades
<i>timestamp</i>	Instante temporal de medida	datetime
<i>datetime</i>	Fecha del valor promedio	datetime
<i>H</i>	Hora del valor promedio	-
<i>overflow</i>	Etiqueta binaria de superación de carga	-
<i>cap</i>	Capacidad del enlace	kW
<i>load</i>	Carga neta (<i>Dif</i>)	kW
<i>dist</i>	Distancia	m
<i>origen_id</i>	Identificador de nodo origen	-
<i>dest_id</i>	Identificador de nodo destino	-
<i>len_origen_tag</i>	Longitud de la etiqueta del nodo origen	-
<i>len_dest_tag</i>	Longitud de la etiqueta del nodo destino	-
<i>modelo</i>	Modelo de topología	-
<i>criterion</i>	Criterio de selección de IDs	-
<i>degree</i>	Grado de conectividad	-
<i>total_balance</i>	Balance de carga global	-
<i>abs_flux</i>	Flujo total de carga en el nodo raíz	-
<i>Diffuse Irradiance</i>	Índice de radiación difusa (DIF)	W/m ²
<i>Plane of Array Irradiance</i>	Índice de radiación en el plano del array (POA)	W/m ²
<i>Ambient Temperature</i>	Temperatura ambiente	C
<i>Cell Temperature</i>	Temperatura de las células solares	C
<i>DC Array Output</i>	Potencia de salida DC del array	W
<i>AC System Output</i>	Potencia de salida AC del sistema	W
<i>Pavg</i>	Potencia consumida	W
<i>Dif</i>	Carga neta calculada	W

Tabla 3.18: Dataset final obtenido por cada instante temporal probado en DEN2NE

4. Desarrollo y evaluación del modelo

En este capítulo, se abordará el desarrollo de los diferentes modelos de predicción de errores y el consecuente entrenamiento de los mismos. Para ello, se tomará como base el conjunto de datos resultante del Capítulo 3 y, específicamente, de la Sección 3.5.3 (ver Tabla 3.18). Como paso previo al desarrollo, se introducirá una Sección de descripción de varias acciones de procesamiento adicional que son necesarias aplicar al conjunto de datos (ver Sección 4.1).

Después, la organización del Capítulo se compone de otras dos partes diferenciadas en función de la naturaleza del aprendizaje que caracteriza a los modelos. Por un lado, se expondrán las técnicas de ML empleadas para resolver la clasificación y predicción de errores (ver Sección 4.2). En el caso de este TFM, se pone el foco en las técnicas RF y SVM, cuyo funcionamiento fue descrito en las Secciones 2.5.1.1 y 2.5.1.2. Por otro lado, se realizará de la misma forma para las técnicas de DL (ver Sección 4.3). En particular, se detallará la configuración y la implementación de varias ANNs. Ambas Secciones comentadas se dividirán internamente, en base a la secuencia de pasos que se requieren para llevar a cabo el desarrollo completo de cada una de las técnicas.

Finalmente, se obtendrán una serie de resultados, que serán analizados y contrastados. En función de su precisión, se podrá determinar de forma concluyente qué modelo aporta una mayor efectividad en el diagnóstico y predicción de errores en una SG.

4.1. Preparación al desarrollo

En primera instancia, antes de proceder a desarrollar cualquier modelo, se debe concatenar en un mismo *dataframe* el contenido de los 12 ficheros resultantes del Capítulo 3. Esta agrupación, como se indicaba en la Sección 3.5.3, se abarca una cantidad total de 1.931.040 filas o intercambios etiquetados sobre los que entrenar los modelos.

A partir de este conjunto de datos, como se pretende resolver un problema supervisado de clasificación, se deben seleccionar las características dependientes (X), que suponen todas las columnas, exceptuando la referente a la etiqueta de error, ‘overflow’, que es la característica independiente (y). Después, se lleva a cabo un paso necesario de tratamiento

de aquellas características que contienen datos categóricos. Particularmente, se observa que la columna ‘modelo’ puede tomar dos valores en formato *string*: ‘barabasi’ o ‘waxman’, en función del modelo de topología empleado.

De la misma forma, ocurre para la fecha, la cual permite 12 posibles cadenas, al haberse probado 12 instantes temporales. Por ello, para poder manejar los datos proporcionados por ambas columnas, se requiere aplicar una transformación a valores numéricos mediante el método *LabelEncoder*. Después, se añaden al *dataframe* las nuevas columnas con los valores codificados y se desechan las originales.

Código 4.1: Codificación de los datos categóricos

```
1 modelo = LabelEncoder().fit_transform(modelo)
2 datetime = LabelEncoder().fit_transform(datetime)
```

Por consiguiente, se procede a analizar el resto de características del conjunto de datos (ver Tabla 3.18) y se toma la decisión de eliminar las columnas que hacen referencia a los diferentes valores de potencia (consumida, producida, carga neta), además de la marca de tiempo. Se aplica este paso por el motivo de que la condición de etiquetado de error en cada intercambio energético viene dada por el exceso de capacidad del enlace. Es decir, como se exponía en la Sección 3.5.1.2, en la ejecución de las simulaciones en DEN2NE se comprueba si cada valor de potencia que se intercambia entre dos nodos supera la capacidad del enlace que los une para activar la etiqueta.

Como consecuencia, no sería correcto entrenar los modelos que se desarrollen en este Capítulo con los datos de potencia, puesto que se estaría reduciendo el análisis a los mismos y, por tanto, la clasificación y la predicción de errores. Esto produciría un *overfitting* o sobreajuste de los modelos y los volvería inefficientes para cumplir los objetivos de este TFM.

Tras los pasos anteriores de tratamiento y limpieza, ya se puede configurar el conjunto de datos y dividirlo en dos subconjuntos: uno se toma como entrada para entrenar el modelo (X_{train} , y_{train}) y otro, se trata como subconjunto de test para evaluar su funcionamiento (X_{test} , y_{test}). Se establece un tamaño de este último del 20% del total del conjunto de datos.

Código 4.2: Codificación de los datos categóricos

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Es importante en este punto estandarizar los datos para asegurar que todas las características poseen la misma escala. Para ello, se aplica el método *StandardScaler* y se llevan a cabo dos acciones dedicadas al ajuste y a la transformación de los datos. Aunque la transformación del escalado se produce sobre los dos subconjuntos, el ajuste viene dado únicamente por los valores de promedio y de desviación estándar del subconjunto de entrenamiento. No se utiliza el de test para ello, puesto que se debe mantener ambos subconjuntos totalmente separados en este proceso. En otros términos, realizar el ajuste a partir de los datos del subconjunto de test podría llevar al *overfitting* o sobreajuste de los modelos. Además, no se produciría una evaluación imparcial porque se estaría operando sobre "predicciones futuras".

Código 4.3: Estandarización de los subconjuntos

```
1 sc = StandardScaler()  
2 X_train = sc.fit_transform(X_train)  
3 X_test = sc.transform(X_test)
```

Una vez se estandarizan los subconjuntos de entrenamiento y de test, se concluye esta etapa de procesamiento adicional para poder comenzar a diseñar los modelos derivados de las técnicas de ML y DL.

4.2. Técnicas de ML

Teniendo en cuenta que los objetivos del presente TFM se basan en la resolución de un problema de clasificación de errores, se toma la decisión de emplear RF y SVM, como técnicas de aprendizaje supervisado. Para desarrollar diversos modelos en base a ambas, se requiere establecer la secuencia de acciones a seguir:

1. Puntuación de características: Se diseña una primera versión o modelo por defecto de la técnica en cuestión para probar su funcionamiento. A partir del mismo, se aprecia las características que presentan mayor importancia en el proceso de clasificación, en base a la aplicación de varios métodos.
2. Optimización de hiperparámetros: Mediante la aplicación del método *Grid Search*, se busca la mejor combinación de hiperparámetros de cada técnica para que el modelo resultante proporcione un rendimiento óptimo.
3. Selección de características: Una vez conocidos los hiperparámetros óptimos, se utilizan varias metodologías de reducción de la dimensionalidad para diseñar los nuevos modelos a partir de un volumen menor de datos.

4. Ejecución del modelo y evaluación de resultados: Se extraen y se analizan los resultados de ejecución. Se validan en base a la precisión obtenida mediante la creación de la matriz de confusión y de la aplicación de la técnica *K-Fold Cross Validation*.

En esta Sección se implementarán dos nuevos *notebooks*, *rf.ipynb* y *svm.ipynb*, los cuales tendrán como base el empleo de diferentes clases y métodos proporcionados por la biblioteca de software de código abierto *scikit-learn* o *sklearn*, además de otras librerías imprescindibles para el manejo de los datos, como son *pandas*, *matplotlib* o *seaborn*. Se dispone de ambos ficheros en el repositorio¹ dedicado al desarrollo de este TFM.

4.2.1. Random Forest (RF)

4.2.1.1. Puntuación de características

Se desarrolla una primera versión del clasificador basado en la técnica de RF, *RandomForestClassifier()* [91], para visualizar las características que más repercuten en el proceso de clasificación de errores. En este modelo por defecto, se construyen un total de 10 estimadores o árboles y se determina el criterio de medición de la impureza a partir de la entropía (ver Sección 2.5.1.1). Después, se aplican dos métodos diferentes para puntuar la importancia que toman las características en el clasificador anterior.

Código 4.4: Clasificador RF por defecto

```
1 classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
2 classifier.fit(X_train, y_train)
```

Por un lado, se estima la importancia de las características del conjunto, a partir del atributo *feature_importances_*, proporcionado por el clasificador. En este caso, el cálculo se realiza a partir de la media y de la desviación estándar de la disminución de la impureza que se produce dentro de cada árbol. Como se representa en la Figura 4.1, se devuelve un array con los valores de importancia relativa asignados a las características y cuyo sumatorio es igual a 1. En este caso, se visualiza una gran incidencia de la distancia, seguida de los parámetros resultantes de DEN2NE, *total_balance* y *abs_flux*, que hacen referencia a la carga que presenta el nodo *gateway* tras el balance y al flujo total de recursos intercambiados en el proceso de distribución energética.

¹<https://github.com/PaulaBartolomeMora/TFM/tree/main/den2ne>

Sin embargo, con este método surge cierto sesgo hacia las características que presentan alta cardinalidad, o en otros términos, una gran cantidad de valores únicos. Esto se debe a que generan nodos de división con mayor profundidad en los árboles, puesto que existen más opciones de separación del conjunto de datos. Por lo tanto, este tipo de características pueden recibir una puntuación más inflada.

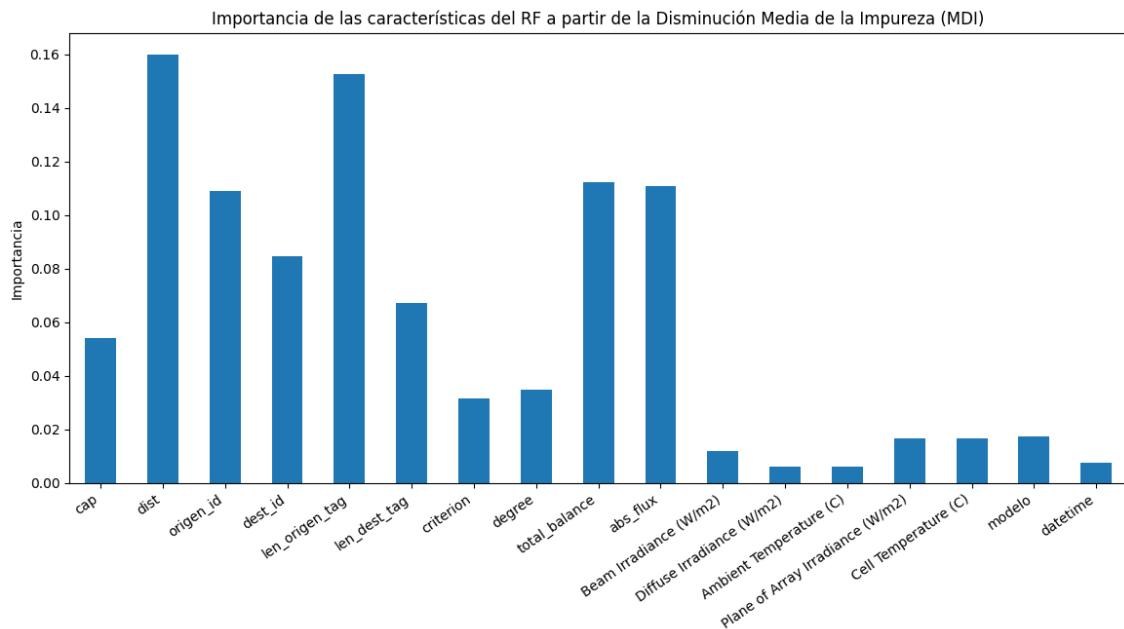


Figura 4.1: Puntuación de características del RF mediante el atributo *feature_importances*

Teniendo esto en cuenta, se decide cuantificar la importancia también por el método de permutación (*permutation_importance*) [92]. Como se puede ver en la Figura 4.2, en este caso, los parámetros resultantes de DEN2NE, *total_balance* y *abs_flux*, siguen presentando puntuaciones altas, pero ahora también, las longitudes de las etiquetas de los nodos y la capacidad del enlace.

En este segundo método, los valores de las características se permutan una a una aleatoriamente y se evalúa en cada iteración los resultados del clasificador. Por tanto, cuando la variación de valores de una característica decrementa de forma considerable la precisión del modelo, puntuá una mayor importancia. Esta técnica es útil porque proporciona una evaluación imparcial de la importancia de las características.

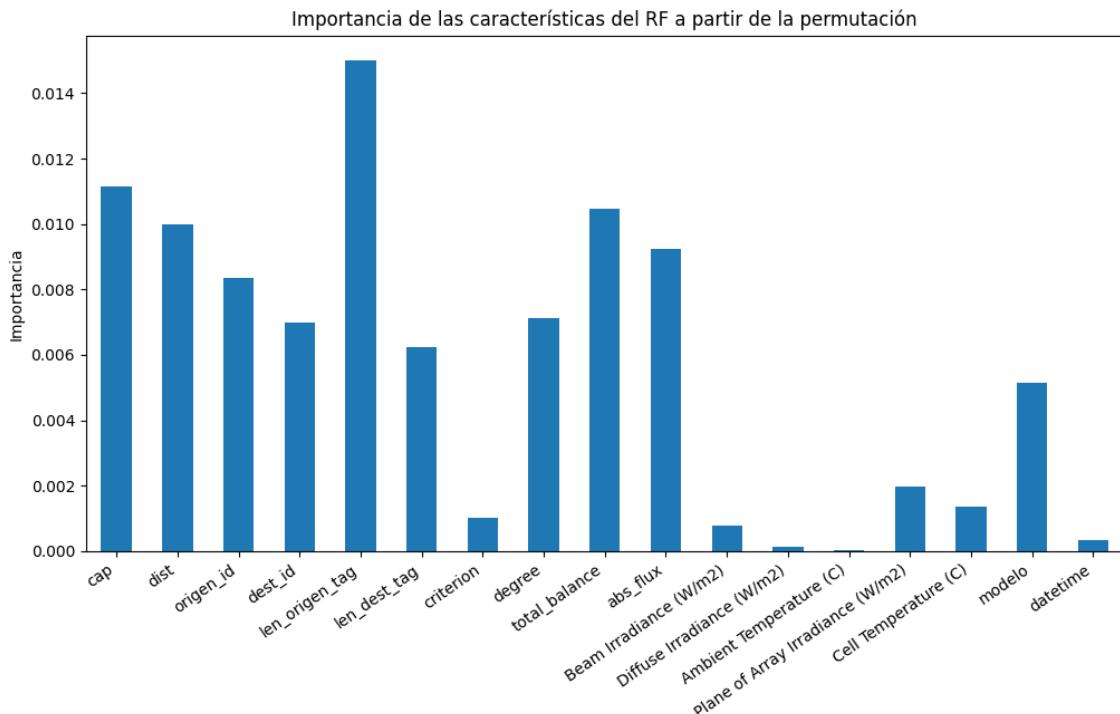


Figura 4.2: Puntuación de características del RF mediante el método *permutation_importance*

4.2.1.2. Optimización de hiperparámetros

Los hiperparámetros de un modelo se definen como los parámetros de configuración o argumentos que son incluidos en el constructor de la clase del estimador o clasificador. En el caso del RF, como ya se había introducido en el diseño del modelo por defecto, se pone el foco en dos hiperparámetros principales: el número de estimadores o árboles y el criterio de medición de la impureza. Para encontrar los valores que construyen el modelo óptimo de RF, es preciso emplear la técnica *Grid Search* [93] y, específicamente, la clase *GridSearchCV()* [94]. Esta realiza una búsqueda exhaustiva a partir de la definición de una cuadrícula de hiperparámetros para el estimador y extrae la combinación de valores que aporta una mayor precisión. Por ello, es necesario primero, definir en un diccionario una serie de valores para los dos hiperparámetros que se pretenden optimizar en el nuevo clasificador.

Código 4.5: Cuadrícula de parámetros RF

```

1 param_grid = {
2     'n_estimators': [10, 25, 50, 75, 100],
3     'criterion': ['entropy', 'gini']
4 }
```

Por consiguiente, se crea el objeto de la clase *GridSearchCV()* y se determina el esquema de validación cruzada (*cross validation*) de este mediante el parámetro *cv*. En este caso, toma un valor de 5 y define el número de pliegues (*folds*) que se van a utilizar para dividir el conjunto de datos y evaluar las combinaciones de los hiperparámetros. De la misma forma, se especifican el resto de parámetros, como la métrica para evaluar el rendimiento del modelo en cada iteración, que viene dada por la precisión global obtenida (*accuracy*).

Una vez creado el objeto, se ajusta al conjunto de datos de entrenamiento. Este proceso, a partir de los atributos *best_score_* y *best_params_*, aporta a la salida cuál es la mejor combinación de parámetros de todas las probadas y qué valor de precisión alcanza la misma. En este caso, se obtiene una precisión del 99,18% con un clasificador basado en 100 estimadores y en el criterio de entropía.

Código 4.6: Construcción del objeto *GridSearchCV()*

```

1 grid_search = GridSearchCV(estimator = classifier,
2                             param_grid = param_grid,
3                             scoring = 'accuracy',
4                             cv = 5,
5                             n_jobs = -1)
6
7 grid_search.fit(X_train, y_train)
```

Sin embargo, para llevar a cabo un análisis en mayor profundidad de los resultados, se hace uso del atributo *cv_results_*, que proporciona un diccionario con toda la información útil de la búsqueda. Principalmente, este análisis se centra en los valores de precisión obtenidos de todos los modelos y en los tiempos promedios que han sido necesarios para ajustar los mismos al conjunto de entrenamiento. Como se puede apreciar en la Tabla 4.1a, se presentan variaciones muy pequeñas en los valores de precisión obtenida (*mean_test_score*) para las diferentes combinaciones de parámetros.

No obstante, en el caso de los tiempos (*mean_fit_time*), como es coherente, sí que se observan grandes diferencias. En la Tabla 4.1b se visualiza cómo se incrementa la duración de la búsqueda proporcionalmente al número de estimadores que se emplea. Esta métrica es importante también tenerla en cuenta para determinar el modelo óptimo, ya que a partir de la aplicación de 25 estimadores, la precisión no mejora de forma considerable. Por lo tanto, tras analizar los resultados, se puede expresar de forma concluyente que la mejor opción de modelo a emplear viene dada por un RF basado en 25 estimadores o árboles y en el criterio de medición de la impureza a partir de la entropía.

Criterio / N° estimadores	Entropía	Gini
10	99,07	99,02
25	99,16	99,14
50	99,16	99,15
75	99,17	99,17
100	99,18	99,16

(a) Precisión (%) (*mean_test_score*)

Criterio / N° Estimadores	Entropía	Gini
10	147	146
25	373	382
50	747	761
75	1072	1002
100	1439	987

(b) Tiempo (s) (*mean_fit_time*)Tabla 4.1: Resultados extraídos del atributo *cv_results_* del *Grid Search* en el RF

4.2.1.3. Selección de características

El proceso de selección de características viene dado por la necesidad de reducir las dimensiones del conjunto de datos y eliminar la información irrelevante o redundante que introduce ruido en el conjunto. En esta Sección, se expone el empleo de tres técnicas diferentes con el fin de realizar posteriormente un análisis comparativo de los resultados que se obtienen tras aplicar cada una de ellas al conjunto de datos (ver Sección 4.2.1.4).

En primer lugar, se introduce la técnica de eliminación recursiva de características con validación cruzada (del inglés *Recursive Feature Elimination with Cross Validation* (RFECV)) [95]. Esta técnica se basa en la ejecución de un proceso iterativo para ir desechando las características que tienen menor influencia en los resultados de precisión, hasta que el rendimiento del modelo deja de mejorar significativamente. Por este motivo, además de proveer a su salida la lista de características más importantes, es capaz de indicar cuál es el número óptimo de características que se debería aplicar para maximizar la precisión en el entrenamiento del modelo, a la vez que se minimiza el volumen de datos en el mismo. En este caso, el RFECV se configura con 5 divisiones (*cv*) del conjunto de datos para llevar a cabo el proceso de evaluación. Además, se determina la eliminación de una de las características disponibles en cada iteración (*step*).

Código 4.7: Aplicación del RFECV

```

1 rfeccv = RFECV(estimator=classifier, step=1, cv=5, scoring='accuracy')
2 rfeccv = rfeccv.fit(X_train, y_train)

```

Código 4.8: Resultados del RFECV

```

1 Nº óptimo de features : 8
2 Selected features : Index(['cap', 'dist', 'origen_id', 'dest_id', 'len_origen_tag', 'len_dest_tag', '↔
    ↪ total_balance', 'abs_flux'], dtype='object')

```

Se puede visualizar el proceso de evaluación del número de características gráficamente en la Figura 4.3, en la cual se representa cómo la precisión del modelo es máxima cuando se emplean 8. Por otro lado, en cuanto a la lista de características proporcionada por el RFECV, es preciso llevar a cabo una comparación con las puntuaciones obtenidas en la Sección 4.2.1.1. Volviendo a las Figuras 4.1 y 4.2, se confirma que, exceptuando ligeras variaciones, la lista de características es coherente con las que presentan mayor grado de importancia.

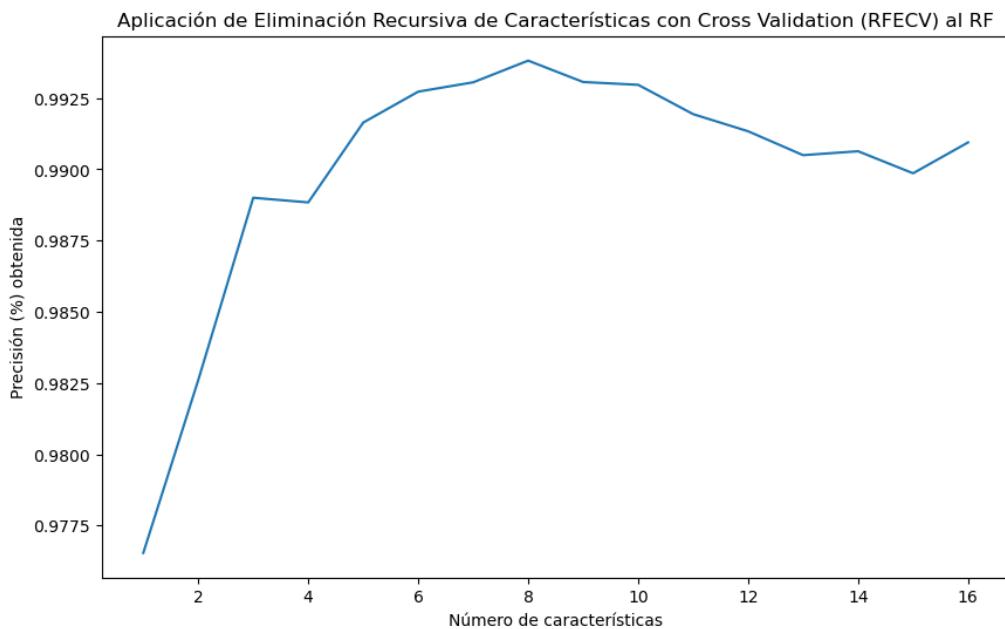


Figura 4.3: Análisis de la precisión del modelo en función del número de características empleadas

Por consiguiente, se entra en el funcionamiento de la segunda técnica a emplear, denominada como Selección Invariante de Características (del inglés *Univariate feature selection*) o, también denominado, *kBest* [96]. En este caso, la operativa incluye un escalado previo de las características, para que sus valores se sitúen en un rango $[0, 1]$.

Después, se declara el objeto de la clase especificada para la técnica en cuestión, *SelectKBest()*, y se determina cuántas características se pretenden seleccionar del total que provee el conjunto. A diferencia del RFECV, para conocer cuál es el número óptimo en la selección invariante de características, es preciso basarse en la prueba y error, puesto que no se proporciona este dato a la salida. Por ello, a modo comparativo, se produce el entrenamiento del conjunto de datos en base a $k=5$ y a $k=8$ características.

Código 4.9: Aplicación del *kBest*

```

1 scaler = MinMaxScaler() # Escalado de las características en el rango [0, 1]
2 kbest = SelectKBest(chi2, k=8)
3 kbest = kbest.fit(scaler.fit_transform(X_train), y_train)

```

Una vez que se extraen aquellas características con mayor puntuación, a partir del atributo *feature_names_in*, se procede a transformar los conjuntos de entrenamiento y de test. Si se compara la lista proporcionada por el atributo anterior con las puntuaciones de las Figuras 4.1 y 4.2, se puede comprobar que esta técnica no es tan precisa como en el caso del RFECV, puesto que se seleccionan algunas características que no presentan tan buenas puntuaciones. Con ello, se podría estimar, a priori, que la implementación de la selección invariante no mejorará los resultados de precisión que se esperan con RFECV.

Código 4.10: Resultados del *kBest* para *k=8*

```

1 Selected features : Index(['cap', 'len_origen_tag', 'len_dest_tag', 'degree', 'abs_flux', 'Beam ↵
    ↪ Irradiance (W/m2)', 'Plane of Array Irradiance (W/m2)', 'Cell Temperature (C)'], dtype=↪
    ↪ 'object')

```

Por último, se introduce una técnica, que no es estrictamente de selección de características, pero que se incluye en esta Sección, puesto que el objetivo que se persigue viene dado por la reducción de la dimensionalidad del conjunto de datos. Se denomina como Análisis de Componentes Principales (del inglés *Principal Component Analysis* (PCA)) [97] y se basa en la aplicación de la técnica matemática de Descomposición en Valores Singulares (del inglés *Singular Value Decomposition* (SVD)). Es decir, su funcionamiento consiste en la búsqueda de las direcciones principales de variación (k vectores) del conjunto de datos para construir una matriz de proyección, que establezca un nuevo espacio de características de k dimensiones.

No obstante, antes de aplicar el ajuste y la transformación al conjunto de datos a partir del PCA, se debe configurar el número de componentes a emplear. Para ello, se utiliza el “método del codo” (del inglés *Elbow Method*) con el fin de identificar el punto o “codo” en el que el aumento del número de componentes no aporta mejoras significativas en el valor de la varianza. Como se representa en la Figura 4.4, en n=4 componentes, se produce el estancamiento más considerable de la varianza, en un valor del 6% aproximadamente. Se puede visualizar que, para el caso de n=2 componentes, también ocurre pero en menor medida. Entonces, a modo comparativo, se considerará aplicar el PCA en función de ambos números.

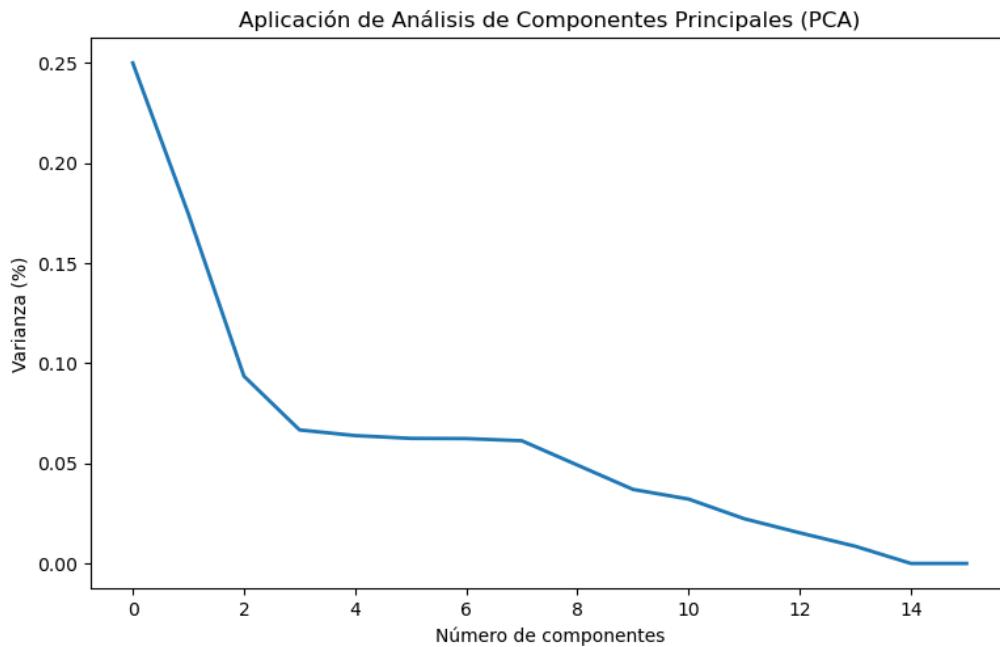


Figura 4.4: Análisis de la varianza en función del número de componentes empleadas en el PCA

4.2.1.4. Ejecución del modelo y evaluación de resultados

Por un lado, como se ha detallado en la Sección 4.2.1.2, el modelo que se considera más adecuado y, por tanto, sobre el que se decide trabajar, viene dado por la configuración de 25 estimadores y el criterio de la entropía. Por otro lado, en la Sección 4.2.1.3, se han expuesto varias propuestas de selección de características o de reducción de dimensiones del conjunto de datos.

Código 4.11: Clasificador RF seleccionado

```

1 classifier = RandomForestClassifier(n_estimators = 25, criterion = 'entropy', random_state = 0)
2 classifier.fit(X_train, y_train)

```

La presente Sección viene dada por la necesidad de aplicar un proceso de evaluación sobre las diferentes implementaciones del modelo de RF seleccionado con el fin de analizar el rendimiento y precisión que proporcionan cada una de ellas. En primera instancia, se va a proceder a ejecutar el modelo de RF, tanto empleando las diferentes opciones de reducción de dimensiones del conjunto de datos, como sin aplicar ninguna de ellas con el objetivo de realizar la comparativa. A modo de proveer una mayor comprensión del entrenamiento que se produce, se incluye la lógica de representación de los estimadores o árboles de decisión del RF, mediante el empleo del método de conversión *export_graphviz*

[98] y la herramienta de generación de grafos, *WebGraphViz*. En la Figura 4.5 se expone uno de los árboles que se construyen en este proceso.

Por consiguiente, una vez ejecutado, se puede llevar a cabo el proceso de evaluación, que consta de dos métodos: la matriz de confusión y el *K-Fold Cross Validation*. La matriz de confusión [99] es un método de gran utilidad cuando se trabaja con técnicas de clasificación binaria, como se produce en este caso, para detectar y predecir los intercambios energéticos con errores. Permite analizar la precisión del modelo mediante la organización de las predicciones en cuatro categorías diferentes (ver Figura 4.6):

- Verdaderos positivos (TP): Predicciones correctas de intercambios con errores.
- Falsos positivos (FP): Predicciones incorrectas de intercambios con errores.
- Verdaderos negativos (TN): Predicciones correctas de intercambios sin errores.
- Falsos negativos (FN): Predicciones incorrectas de intercambios sin errores.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figura 4.6: Matriz de confusión [101]

A través de las categorías definidas en la matriz, se pueden obtener cuatro métricas de evaluación:

- *Accuracy*: Mide la proporción de predicciones correctas en relación con el total y evalúa el rendimiento general de un modelo de clasificación.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

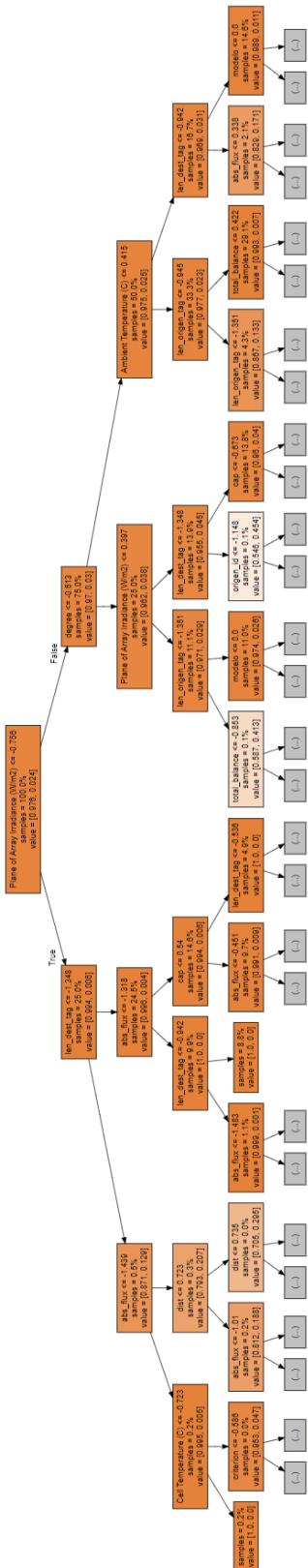


Figura 4.5: Ejemplo gráfico de un estimador o árbol del RF [100]

- *Precision*: Mide la proporción de verdaderos positivos entre todas las instancias clasificadas como positivas.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4.2)$$

- *Recall*: Mide la proporción de verdaderos positivos identificados correctamente entre todas las instancias que son realmente positivas.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.3)$$

- *F1 Score*: Combina las dos métricas anteriores, tomando en cuenta tanto los falsos positivos como los falsos negativos.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.4)$$

Código 4.12: Implementación de la matriz de confusión

```
1 cm = confusion_matrix(y_test, y_pred)
2 accuracy_score(y_test, y_pred)
```

Por tanto, con la aplicación de la matriz de confusión, se extraen los resultados expuestos en la Tabla 4.2. En la misma, se puede visualizar la cantidad de predicciones, en función de cada una de las categorías de la matriz y los valores obtenidos del cálculo de las métricas anteriores. A priori, si solo se pone el foco en la precisión global del modelo (*Accuracy*), las opciones que proveen valores más altos son la que no aplica ningún método de reducción de dimensiones y que por tanto, opera con todas las características del conjunto, y la que emplea la técnica RFECV.

En el caso de la precisión de los valores positivos (*Precision*) o, en otros términos, de las instancias que se predicen como erróneas, vuelve a ocurrir de la misma manera. A pesar de que para la métrica *Accuracy* los valores de todas las opciones son relativamente parecidos, para *Precision* se visualizan grandes diferencias. Por ejemplo, el empleo de un PCA con n=2 provee un buen valor de precisión global (97,55%), pero no tiene un buen rendimiento en la clasificación de los errores (12,52%). En consecuencia, el valor del *Recall* también es muy pequeño en este caso (0,67%). El resto de opciones, exceptuando las dos primeras, ya mencionadas anteriormente, presentan también porcentajes bajos, suponiendo un coste alto de predicción de errores.

<i>Matriz de confusión</i>	<i>TN</i>	<i>FP</i>	<i>FN</i>	<i>TP</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
<i>Sin aplicar</i>	376745	399	2332	6732	99,29	94,40	74,27	83,16
<i>RFECV</i>	376699	445	1717	7347	99,44	94,28	81,05	87,17
<i>kbest (n=5)</i>	376555	589	7927	1137	97,79	65,97	12,55	21,08
<i>kbest (n=8)</i>	375027	2117	6583	2481	97,74	53,95	27,37	36,32
<i>PCA (n=2)</i>	376718	426	9003	61	97,55	12,52	00,67	01,27
<i>PCA (n=4)</i>	376684	460	7015	2049	98,06	81,66	22,60	35,41

Tabla 4.2: Resultados de aplicación de la matriz de confusión al RF

De la misma forma, se puede visualizar que para la métrica *F1 Score* se esperan resultados relativamente cercanos al *Recall*. Al considerarse tanto los falsos positivos, como los falsos negativos, produce que la obtención de un porcentaje significativamente pequeño indique una baja *Precision* y *Recall* conjuntamente.

Por otro lado, en cuanto al método *K-Fold Cross Validation* [102], se emplea la misma operativa que en el *Grid Search*. Como su nombre indica, se basa en un esquema de validación cruzada (*cross validation*) y se divide el conjunto de datos en un total de 5 pliegues para evaluar el rendimiento del modelo. En la Tabla 4.3, se determinan los resultados de su aplicación y, como se puede visualizar, los valores de precisión global son prácticamente iguales a los obtenidos anteriormente en la matriz de confusión.

Por tanto, mediante el empleo de las dos técnicas y el análisis expuesto, se puede confirmar definitivamente que las opciones con mejores resultados hacen referencia a la que no se aplica ningún método de reducción de dimensiones y a la que emplea RFECV, siendo esta última la que proporciona un rendimiento óptimo.

Código 4.13: Implementación del *K-Fold Cross Validation*

```
1 accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 5)
```

<i>K-Fold Cross Validation</i>	<i>Accuracy (%)</i>	<i>Standard Deviation (%)</i>
<i>Sin aplicar</i>	99,30	0,02
<i>RFECV</i>	99,47	0,02
<i>kbest (n=5)</i>	97,80	0,02
<i>kbest (n=8)</i>	97,79	0,01
<i>PCA (n=2)</i>	97,35	0,01
<i>PCA (n=4)</i>	98,04	0,00

Tabla 4.3: Resultados de aplicación del *K-Fold Cross Validation* al RF

4.2.2. Support Vector Machines (SVM)

4.2.2.1. Puntuación de características

Para la técnica de SVM, también se procede a desarrollar una primera versión del clasificador, con el fin de visualizar la importancia que toman cada una de las características en el mismo. En este caso, se trata de la clase *SVC()* [103] y se inicializa un objeto de la misma, definiendo por defecto un kernel radial o gaussiano (RBF). Esto se debe a que se está trabajando con un conjunto de datos no linearmente separables y que resulta complejo de operar si no se aumenta la dimensionalidad del espacio (ver Sección 2.5.1.2).

De la misma forma, *SVC()* determina valores por defecto de los hiperparámetros *C* y *gamma*, que hacen referencia respectivamente, a la regularización del modelo y al impacto que produce cada instancia de entrenamiento en el proceso de clasificación.

Código 4.14: Clasificador SVM por defecto

```

1 classifier = SVC(kernel = 'rbf', random_state = 0) #por defecto C=1, gamma='scale' o 'auto'
2 classifier.fit(X_train, y_train)

```

En este paso, cabe destacar que el proceso de ejecución y entrenamiento del clasificador del SVM sobre el conjunto de datos presenta una duración mucho mayor que el RF. Una vez finaliza, se le aplican dos métodos de puntuación de características.

Por un lado, se estima la importancia de las características a partir de la distancia que tienen hacia los vectores de soporte. En este caso, es preciso basarse en los atributos *support_vectors_* y *dual_coef_*, que proporcionan la información sobre los vectores soporte que se han definido en el SVM y los multiplicadores de *Lagrange* asociados.

El resultado se expone en la Figura 4.7, en la cual se visualiza cómo las longitudes de las etiquetas origen y destino inciden en la clasificación significativamente, ya que el resto de características tienen puntuaciones mucho más bajas.

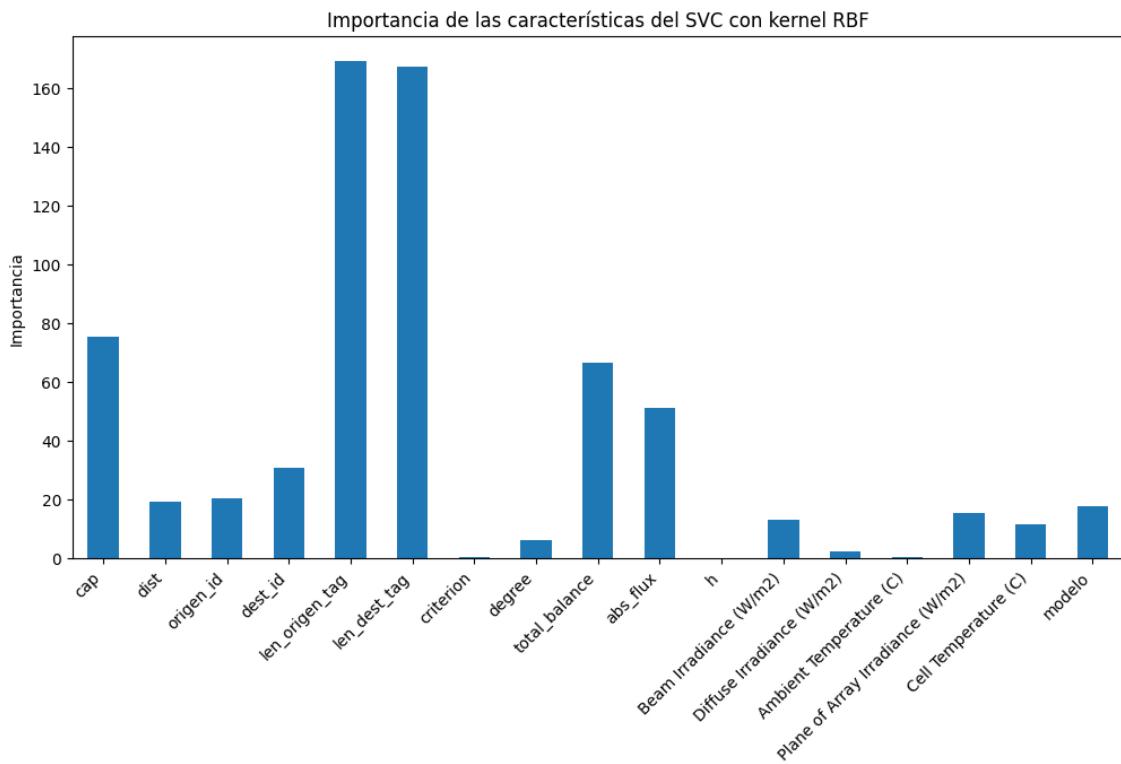


Figura 4.7: Puntuación de características del SVM mediante los atributos *support_vectors_* y *dual_coef_*

Por otro lado, al igual que se ha expuesto para el RF (ver Sección 4.2.1.1), se incluye el análisis a partir del método de la permutación (*permutation_importance*). En este caso, las longitudes de las etiquetas siguen siendo predominantes, pero las características que hacen referencia a la capacidad y al flujo total de energía resultante de DEN2NE (*abs_flux*) también presentan valores de puntuaciones a considerar.

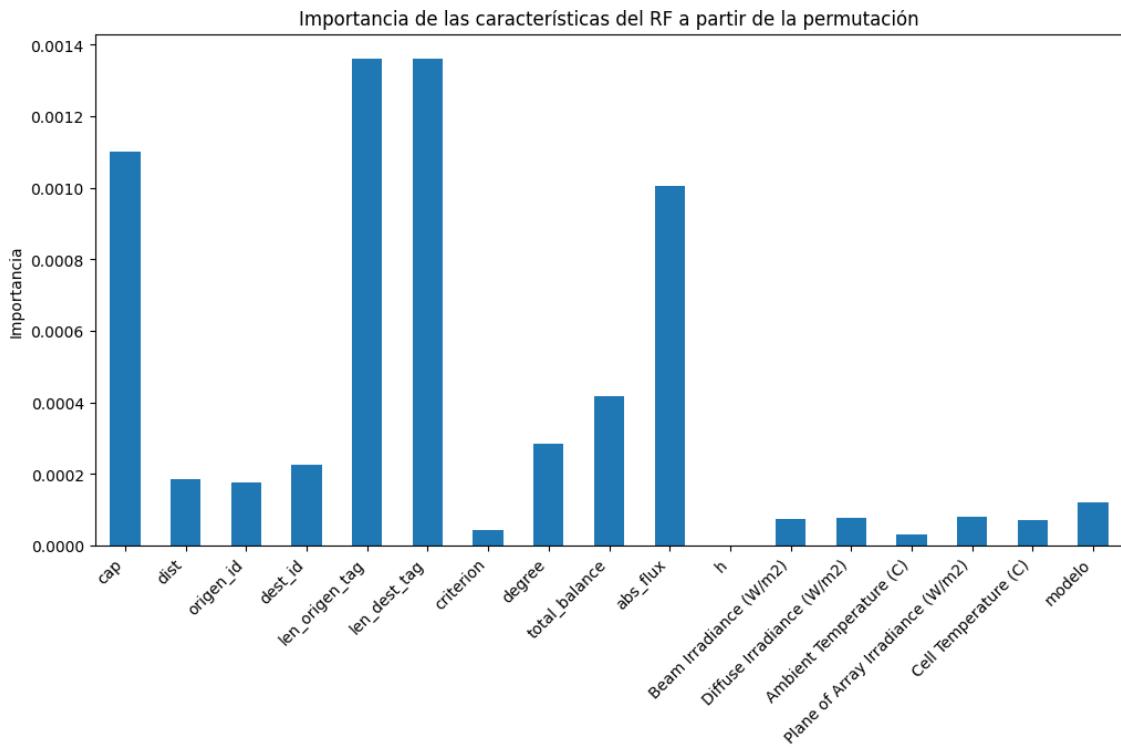


Figura 4.8: Puntuación de características del SVM mediante el método *permutation_importance*

4.2.2.2. Optimización de hiperparámetros

Para modelar un SVM óptimo se centra el estudio en dos hiperparámetros: el tipo de kernel a aplicar y el parámetro de regularización C . En el caso de γ , mencionada anteriormente, es indiferente su configuración, puesto que los datos del conjunto han sido previamente escalados (ver Sección 4.1) y las dos opciones posibles de γ ('scale' o 'auto') únicamente se diferencian entre sí en la aplicación del valor de la varianza de los datos en el cálculo del parámetro.

Código 4.15: Cuadrícula de parámetros SVM

```

1 param_grid = {
2     'C': [0.25, 0.5, 0.75, 1],
3     'kernel': ['poly', 'rbf', 'sigmoid'] }
```

Después, se inicializa un objeto de la clase *GridSearchCV()* y se configura de la misma forma que se detallaba en la Sección 4.2.1.2 para el RF. Sin embargo, teniendo en cuenta la gran duración del entrenamiento del modelo SVM en el paso anterior (ver Sección 4.2.2.1), se debe cuantificar primero cuánto tiempo supone ejecutar todas las combinaciones de hiperparámetros definidas en el método *Grid Search*. Este cálculo se realiza en función

de los pliegues configurados ($cv=5$) y del número de *cores*, por lo que se estima el tiempo total de búsqueda de la combinación óptima en 349,4 horas.

La ejecución indica en los atributos *best_score_* y *best_params_* que el caso óptimo es un SVM basado en kernel radial o gaussiano (RBF) con un parámetro de regularización C igual a la unidad, con el que se alcanza una precisión del 97,78%. No obstante, se debe llevar a cabo un análisis más detallado de los resultados de rendimiento de las combinaciones probadas, a partir del atributo *cv_results_*. Este diccionario aporta los valores promedio de precisión obtenida (*mean_test_score*) como se muestra en la Tabla 4.4a. Se pueden destacar variaciones en la precisión al aplicar los distintos tipos de kernel al clasificador, siendo el kernel sigmoide el que peores resultados proporciona. Sin embargo, el empleo de diferentes parámetros de regularización C no implica grandes cambios en la precisión y, en el caso del kernel polinomial, es indiferente y se obtienen siempre los mismos valores.

De la misma forma, se extrae también de *cv_results_* el tiempo de entrenamiento (*mean_fit_time*) que se ha dedicado para probar cada una de las combinaciones. Como se puede ver en la Tabla 4.4b, ahora el valor que toma el parámetro de regularización C sí que incide de forma notable en los valores y aumenta de forma proporcional en el caso del kernel polinomial. Se puede expresar que si se desea emplear este tipo de kernel no sería adecuado configurar un parámetro de regularización alto, ya que el rendimiento empeora.

<i>Kernel / C</i>	<i>poly</i>	<i>rbf</i>	<i>sigmoid</i>
0,25	97,65	97,66	95,96
0,5	97,65	97,71	95,94
0,75	97,65	97,75	95,82
1	97,65	97,78	95,81

(a) Precisión (%) (*mean_test_score*)

<i>Kernel / C</i>	<i>poly</i>	<i>rbf</i>	<i>sigmoid</i>
0,25	21030	13876	14659
0,5	28807	10277	18695
0,75	42183	9328	12235
1	47460	9446	12869

(b) Tiempo (s) (*mean_fit_time*)**Tabla 4.4:** Resultados extraídos del atributo *cv_results_* del SVM

Sin embargo, mediante el análisis de los tiempos, se confirma que la opción más adecuada a emplear es el caso óptimo descrito a priori por el atributo *best_params_*, ya que proporciona un tiempo de entrenamiento relativamente bajo respecto al resto de combinaciones.

4.2.2.3. Selección de características

Para el modelo de SVM, se procede a aplicar las mismas tres técnicas de reducción de la dimensionalidad del conjunto de datos que se utilizaron en el RF (ver Sección 4.2.1.3): RFECV, *kbest* y PCA. Sin embargo, en el caso de la primera, referente a la eliminación recursiva mediante validación cruzada, se encuentra con la problemática de que la clase del clasificador, *SVC()*, no dispone de los atributos necesarios para su implementación (*coef_*, *feature_importances_*). Entonces, se tiene que descartar su uso y seguir con la aplicación del resto de técnicas.

En cuanto a la Selección Invariante de Características o *kbest*, se mantiene la operativa realizada para el modelo de RF. Como se comentaba en la Sección 4.2.1.3, el número de características a seleccionar en esta técnica venía dado por la necesidad de realizar múltiples pruebas para poder observar con qué cantidad se proporciona una mayor precisión. Para simplificar este proceso, se aplicaban dos valores de *k* (*k=5* y *k=8*), siendo el segundo el dado como resultado de la técnica RFECV. En este caso, como dicha técnica no se puede implementar, se toma la decisión de escoger los mismos valores que se configuraron en el modelo de RF.

Por consiguiente, para emplear la técnica PCA, se repite el proceso descrito en la Sección 4.2.1.3, aplicando la técnica para reducir las dimensiones del conjunto de datos a *n=2* y *n=4* componentes.

4.2.2.4. Ejecución del modelo y evaluación de resultados

Tras desarrollar el proceso de optimización de hiperparámetros del clasificador, se procede a evaluar el rendimiento del modelo SVM utilizando las diferentes técnicas de reducción de las dimensiones del conjunto de datos descritas anteriormente. Es importante volver a recalcar, antes de analizar los resultados, el significativo coste computacional y tiempo asociado con el entrenamiento del clasificador. En este caso, se ejecutan 5 pruebas aplicando en cada una el método de la matriz de confusión y del *K-Fold Cross Validation*, lo que resulta en una duración total de 325 horas.

Para analizar los resultados de evaluación a partir de la matriz de confusión, se proporciona la Tabla 4.5. Observando la misma, se puede destacar a simple vista que en los dos casos de aplicación del PCA se obtienen valores indefinidos en las métricas de *Precision* y *F1 Score*. Esto se debe a que el modelo de SVM no funciona correctamente y no logra realizar ninguna predicción de instancias positivas. Por tanto, aunque se obtiene un valor de precisión global (*Accuracy*) bueno, no es un modelo realmente eficiente. Respecto a esta métrica, si se visualizan las demás opciones, se obtienen también valores en torno

al 97%. En el caso de la precisión de detección de las instancias positivas (*Precision*), la opción más favorable es la ejecución del modelo de SVM sin emplear ninguna técnica de reducción de dimensiones (89,32%). Es decir, al aplicar *kbest* al conjunto de datos, se reduce la eficiencia cuanto menor es el número de características *k* configurado (65,25%).

Por otro lado, la métrica *Recall* presenta en todas las opciones valores bajos. Al referirse a la sensibilidad del modelo o, en otros términos, a la proporción de las instancias positivas que se identifican correctamente, el coste de predicción de errores se vuelve alto de forma general para el modelo. Con estos valores y los dados por *F1 Score*, se determina de una forma concluyente que el SVM no es lo suficientemente eficiente para detectar las instancias positivas.

<i>Matriz de confusión</i>	<i>TN</i>	<i>FP</i>	<i>FN</i>	<i>TP</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
<i>Sin aplicar</i>	377009	76	8489	634	97,23	89,32	6,95	12,83
<i>kbest (n=5)</i>	376487	598	8002	1121	97,11	65,25	12,29	20,73
<i>kbest (n=8)</i>	376742	343	8274	849	97,05	71,19	9,30	16,46
<i>PCA (n=2)</i>	377085	0	9123	0	97,61	-	0	-
<i>PCA (n=4)</i>	377085	0	9123	0	97,61	-	0	-

Tabla 4.5: Resultados de aplicación de la matriz de confusión al SVM

En cuanto a los resultados de aplicación del método *K-Fold Cross Validation*, se puede comprobar en la Tabla 4.6 que se obtienen valores de precisión parecidos a los dados anteriormente. En este caso, con mínimas diferencias, se presenta como mejor opción la aplicación de la técnica *kbest* con 5 características con un 97,80%.

Entonces, tras completar el proceso de evaluación del modelo mediante ambos métodos, se puede expresar que se perciben generalmente mejores resultados para el RF que para el SVM. Es decir, aunque los resultados de aplicar *K-Fold Cross Validation* en ambos modelos no proporcionan precisiones significativamente diferentes entre sí (ver Tablas 4.3 y 4.6), si se comparan las métricas calculadas a partir de las matrices confusión (ver Tablas 4.2 y 4.5), el rendimiento que aporta el RF es mucho mayor.

<i>K-Fold Cross Validation</i>	<i>Accuracy (%)</i>	<i>Standard Deviation (%)</i>
<i>Sin aplicar</i>	97,79	0,01
<i>kbest (n=5)</i>	97,80	0,01
<i>kbest (n=8)</i>	97,79	0,01
<i>PCA (n=2)</i>	97,76	0,00
<i>PCA (n=4)</i>	97,76	0,00

Tabla 4.6: Resultados de aplicación del *K-Fold Cross Validation* al SVM

4.3. Técnicas de DL

De la misma forma que se ha realizado para las técnicas de ML, se debe indicar la secuencia de acciones a seguir en esta Sección referente al DL. No obstante, antes que nada, es necesario volver a la Figura 2.25, expuesta en la Sección 2.5.2. En la misma, se podían apreciar las diferencias estructurales que presentaban las técnicas de ML y de DL entre sí. Al radicar principalmente en el tratamiento de las características del conjunto de datos, ahora para desarrollar los diferentes modelos de ANNs, se debe seguir la secuencia de acciones comentada en la Sección 4.2, pero eliminando los pasos de puntuación y selección de las características (pasos 1 y 3).

El desarrollo vendrá dado por la implementación de los *notebooks*, *ann.ipynb* y *mlp.ipynb*. Es preciso destacar que, en el caso del primero, se empleará como base el módulo de redes neuronales *keras*, que está integrado en la librería *TensorFlow* [104] y que se trata de la API de alto nivel predeterminada a partir de la versión 2.0 de la misma. De forma adicional, en ambos se utilizarán clases y métodos de *scikit-learn* o *sklearn* y se importarán el resto de librerías necesarias para el tratamiento y representación de datos. Se dispone de ambos ficheros en el repositorio² dedicado al desarrollo de este TFM.

4.3.1. Redes Neuronales Artificiales (ANN)

Antes de entrar en detalle en la optimización de hiperparámetros de la ANN, se debe indicar que esta Sección se divide en dos fases diferenciadas: primero, se centra la aplicación del método *Grid Search* sobre el modelo de MLP proporcionado por la librería *sklearn* y, después, los resultados obtenidos de la búsqueda se utilizan para configurar una nueva ANN optimizada, basada en el módulo *keras*. Esta secuencia de pasos se establece a modo de simplificar la comprensión y de justificar y comparar los resultados que se obtienen para ambas versiones de ANNs.

²<https://github.com/PaulaBartolomeMora/TFM/tree/main/den2ne>

4.3.1.1. Optimización de hiperparámetros y ejecución del modelo de ANN de *sklearn*

En primera instancia, se determinan los hiperparámetros que se van a estudiar del MLP [105]. Estos hacen referencia a la configuración de capas ocultas y al número de neuronas que puede tener cada capa (*hidden_layer_sizes*), a la función de activación que se aplica (*activation*) y al algoritmo de optimización de los pesos de la red durante el proceso de entrenamiento (*solver*).

Código 4.16: Cuadrícula de parámetros MLP

```

1 param_grid = {
2     'hidden_layer_sizes': [(5,), (8,), (10,), (50,), (100,), (5, 5), (8, 8), (10, 10), (50, 50)],
3     'activation': ['relu', 'tanh'],
4     'solver': ['sgd', 'adam']
5 }
```

En este caso, se configura para la búsqueda un modelo por defecto de MLP, en el que se aplican un máximo de 100 iteraciones por cada combinación a probar. De la misma forma, para no introducir latencias innecesarias ni un sobreentrenamiento que perjudique a los resultados de clasificación, se determina una finalización temprana del entrenamiento cuando se produzcan una cantidad de iteraciones seguidas sin mejoras significativas. En este caso, se deja el valor por defecto, que es 10 iteraciones y se modifica la tolerancia a un valor de 0.00001.

Código 4.17: Clasificador MLP por defecto

```

1 mlp = MLPClassifier(max_iter=100, verbose=True, early_stopping=True, tol=0.00001)
```

La creación del objeto de la clase *GridSearchCV()*, con una configuración de 5 pliegues (*cv=5*), y su entrenamiento sobre el MLP anterior en un equipo de 32 procesadores, se estima en una duración de 2,34 horas. Cuando se lleva a cabo este proceso, se puede comprobar en el diccionario de resultados (*cv_results_*) y, en particular, en los valores de precisión de la variable *mean_test_score*, que el empleo del algoritmo de optimización del gradiente descendiente estocástico (SGD) aporta mejores resultados que el *adam*.

En la Tabla 4.7 se pueden encontrar varias combinaciones de hiperparámetros que optimizan el rendimiento, con un valor de precisión del 97,6%. Sin embargo, también es importante poner el foco en los valores de desviación típica que se consiguen y en la duración del entrenamiento que supone cada opción para determinar cuál es la combinación de hiperparámetros más adecuada para el MLP.

En el caso de la desviación típica, como se observa en la Tabla 4.8, no se producen grandes diferencias de valores entre las configuraciones en cuestión, por lo que, a priori, no es un factor determinante a tener en cuenta en la selección. Sin embargo, en el caso de los tiempos sí existen variaciones, las cuales se deben al número de iteraciones que son necesarias para alcanzar la convergencia para cada versión de MLP y a la duración que supone cada iteración.

En la Tabla 4.9, se puede visualizar que, cuanto mayor sea el número de neuronas especificado, mayor será la latencia introducida. Esto presenta también, cierta correlación con los valores de la Tabla 4.7, ya que cuando el proceso de entrenamiento es relativamente largo, es posible que se produzca un sobreentrenamiento o *overfitting* del modelo de MLP y que la precisión obtenida sea más baja. Por ello, es importante configurar la tolerancia de optimización de forma correcta y detener el entrenamiento cuando se detecte la convergencia del modelo (*early stopping*).

Por lo tanto, tras analizar los resultados, se expone de forma concluyente que la combinación de hiperparámetros más adecuada y que debería ser la configurada en el modelo de MLP, es la que determina una estructura de dos capas ocultas de 5 neuronas, una función de activación *relu* y el algoritmo de optimización SGD. En este caso, como se puede apreciar en las Figuras 4.9 y 4.10 el proceso de entrenamiento converge en la iteración 41 con una precisión del 97,66% y una función de pérdidas con valor 0,0712.

<i>Solver</i>	<i>adam</i>		<i>sgd</i>	
<i>Función de activación / Capas ocultas</i>	<i>relu</i>	<i>tanh</i>	<i>relu</i>	<i>tanh</i>
(5,)	90,17	97,61	97,66	97,54
(8,)	92,03	89,73	97,65	97,65
(10,)	87,00	89,90	97,48	89,76
(50,)	86,89	89,82	89,53	97,65
(100,)	90,95	89,71	84,56	89,72
(5, 5)	89,80	89,72	97,66	97,65
(8, 8)	88,49	89,72	89,89	97,65
(10, 10)	81,68	89,70	90,30	97,65
(50, 50)	81,91	82,04	97,66	89,73

Tabla 4.7: Resultados de precisión (%) (*mean_test_score*) extraídos del atributo *cv_results_* del MLP

<i>Solver</i>	<i>adam</i>		<i>sgd</i>	
	<i>relu</i>	<i>tanh</i>	<i>relu</i>	<i>tanh</i>
(5,)	14,75	0,10	0,01	0,22
(8,)	11,33	15,84	0,00	0,00
(10,)	15,54	15,51	0,33	15,78
(50,)	15,58	15,74	11,09	0,00
(100,)	9,40	15,90	16,65	15,87
(5, 5)	15,83	15,87	0,01	0,00
(8, 8)	12,19	15,87	15,52	0,00
(10, 10)	19,77	15,89	14,71	0,00
(50, 50)	19,52	19,20	0,01	15,85

Tabla 4.8: Resultados de desviación típica (%) (*std_test_score*) extraídos del atributo *cv_results_* del MLP

<i>Solver</i>	<i>adam</i>		<i>sgd</i>	
	<i>relu</i>	<i>tanh</i>	<i>relu</i>	<i>tanh</i>
(5,)	88	37	63	35
(8,)	96	43	35	36
(10,)	94	40	56	43
(50,)	157	168	143	70
(100,)	236	223	170	131
(5, 5)	136	50	52	44
(8, 8)	159	54	69	46
(10, 10)	149	55	54	48
(50, 50)	333	319	166	155

Tabla 4.9: Resultados de tiempo (s) (*mean_fit_time*) extraídos del atributo *cv_results_* del MLP

Código 4.18: Clasificador MLP óptimo

```
1 mlp = MLPClassifier(max_iter=100, verbose=True, early_stopping=True, tol=0.00001, ↵
    ↵ hidden_layer_sizes=(5, 5), activation='relu', solver='sgd')
```

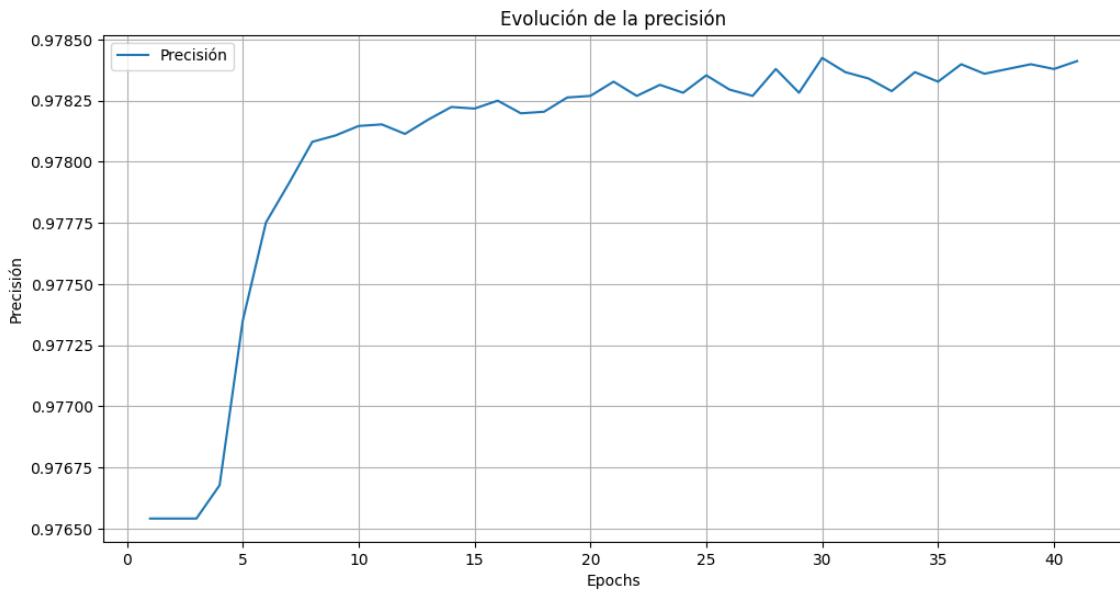


Figura 4.9: Representación del valor de precisión en función de las iteraciones para el modelo de MLP escogido

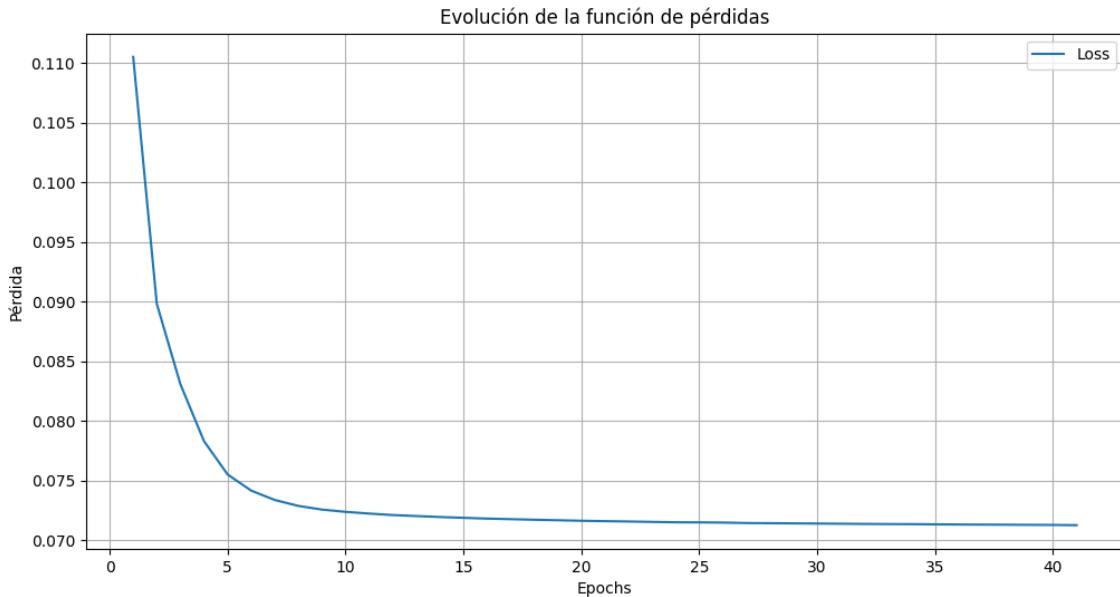


Figura 4.10: Representación del valor de la función de pérdidas en función de las iteraciones para el modelo de MLP escogido

4.3.1.2. Ejecución del modelo de ANN de keras y evaluación de resultados

Una vez seleccionada la combinación de hiperparámetros que produce un rendimiento óptimo del MLP, el siguiente paso consiste en aplicar la configuración en cuestión a un nuevo modelo de ANN. Como se había introducido al comienzo de la Sección 4.3.1, el objetivo es realizar una comparativa entre dos ANNs proporcionadas por distintas librerías, como son en este caso *sklearn* y el módulo *keras* de *tensorflow*, y así, comprobar que los resultados obtenidos de aplicar el método *Grid Search* anteriormente son coherentes.

Teniendo esto en cuenta, se define la estructura del modelo de ANN de *keras* [106] con dos capas ocultas de 5 neuronas y una capa de una neurona a la salida, puesto que se trabaja con un conjunto de datos con etiquetas binarias. Además, es necesario especificar a la entrada el número de características (*input_shape*), ya que será igual al número de entradas de la red neuronal. La estructura del modelo definido se representa gráficamente en la Figura 4.11 mediante el uso de la herramienta online proporcionada por *TensorFlow*.

Código 4.19: Definición del modelo de ANN de Keras

```

1 model = keras.Sequential([
2     keras.layers.Dense(5, input_shape=(X.shape[1],), activation='relu'),
3     keras.layers.Dense(5, activation='relu'),
4     keras.layers.Dense(1, activation='sigmoid')
5 ])

```

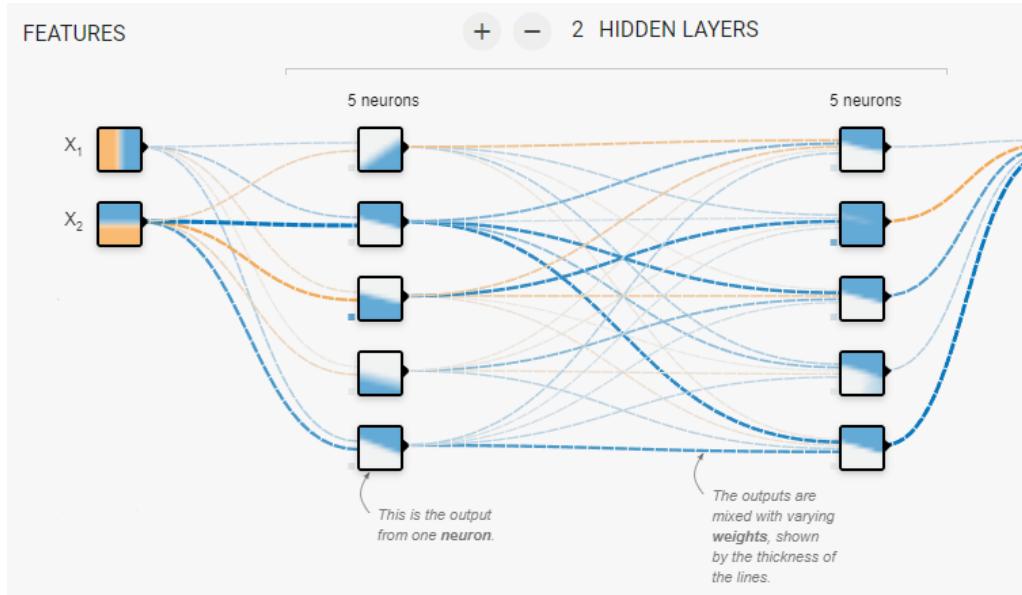


Figura 4.11: Representación de la estructura de capas de la ANN [104]

Por consiguiente, se compila la red configurando el algoritmo de optimización SGD. También, la función de pérdidas binaria (*binary_crossentropy*), cuyo valor vendrá dado por la entropía cruzada media entre los positivos que se predicen y los que realmente lo son. Después, se inicia el entrenamiento para un total de 100 iteraciones y se activa la finalización temprana [107] del proceso, en el caso de acontecer 10 iteraciones seguidas sin mejoras significativas (0.00001) en la función de pérdidas.

Código 4.20: Entrenamiento del modelo de ANN de Keras

```
1 model.compile(optimizer = 'sgd', loss = 'binary_crossentropy', metrics = ['accuracy'])
2 History = model.fit(X_train, y_train, epochs=100, verbose=1, callbacks=[keras.callbacks.EarlyStopping(←
→ monitor='val_loss', patience=10, verbose=1, min_delta=0.00001)])
```

De la misma forma que en el modelo anterior, se almacenan los valores de precisión y de pérdidas de cada iteración y se representan en forma de gráfica lineal en las Figuras 4.12 y 4.13. Como se puede ver, en este caso el entrenamiento del modelo de ANN de *keras* no finaliza antes de llegar a las 100 iteraciones como ocurría con el MLP. Sin embargo, los valores de la función de pérdidas y de precisión que se adquieren son muy similares respecto al modelo anterior, siendo de 0,068 y del 97,87%, respectivamente.

Una vez se han definido y entrenado los dos modelos de ANN anteriores, se introduce una etapa final de evaluación y validación mediante dos técnicas. Por un lado, se aplica la matriz de confusión, al igual que se realizaba para los modelos de ML y, por otro lado, se hace empleo de los propios métodos de evaluación que ofrecen ambas ANNs. Como en esta Sección, enfocada al DL, no se lleva a cabo una etapa de selección de características, la evaluación se ciñe a la comparación de resultados de ambos modelos de ANNs.

En la Tabla 4.10, se puede apreciar la similitud de los resultados, obteniéndose un mismo valor de precisión global (*Accuracy*) para los dos modelos implementados (97,84%). No obstante, si se pone el foco en el resto de métricas, se percibe una efectividad en la detección de errores (*Recall*) ligeramente mayor para la ANN de *sklearn* (14,98%), mientras que la ANN de *keras* produce una mayor confiabilidad (*Precision*) y, por tanto, una menor probabilidad de falsos positivos (72,17%).

Por consiguiente, en la Tabla 4.11 se exponen los resultados de aplicar los métodos *score* y *evaluate*, proporcionados por *sklearn* y *keras*, respectivamente. En el primer caso, solo se puede obtener el valor de precisión del modelo, mientras que en el segundo se aportan también las pérdidas. Como es de esperar, los resultados siguen siendo similares para ambas ANNs y, por tanto, se puede expresar que el estudio realizado en esta Sección es coherente. De forma concluyente, aunque las diferencias entre ambos modelos son mínimas, como se

prioriza la efectividad en la detección de positivos, se escogería el modelo desarrollado con la librería *keras*.

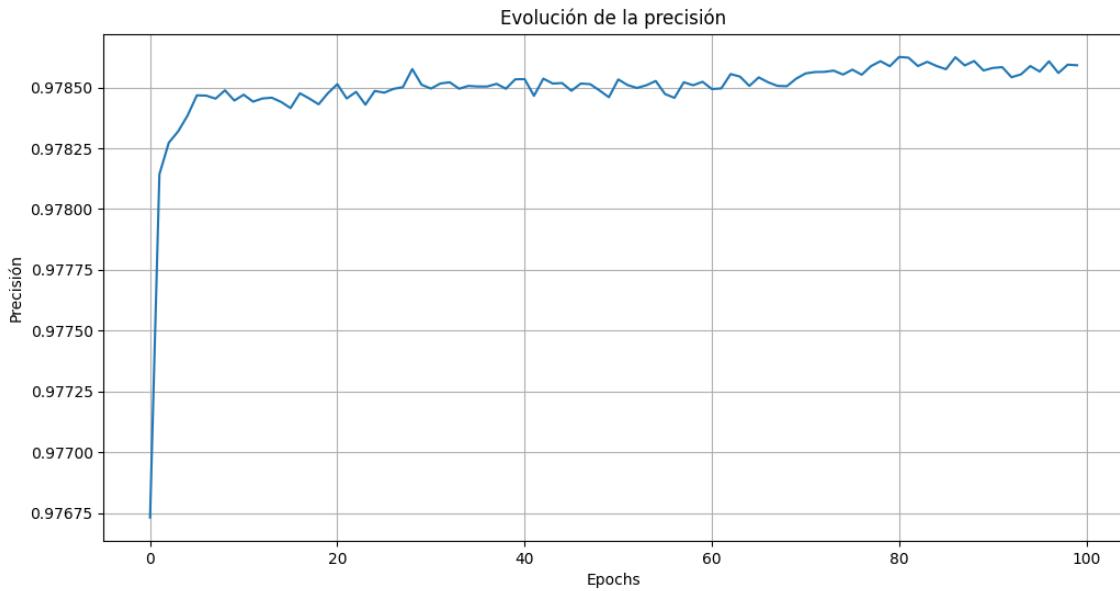


Figura 4.12: Representación del valor de precisión en función de las iteraciones para el modelo de ANN

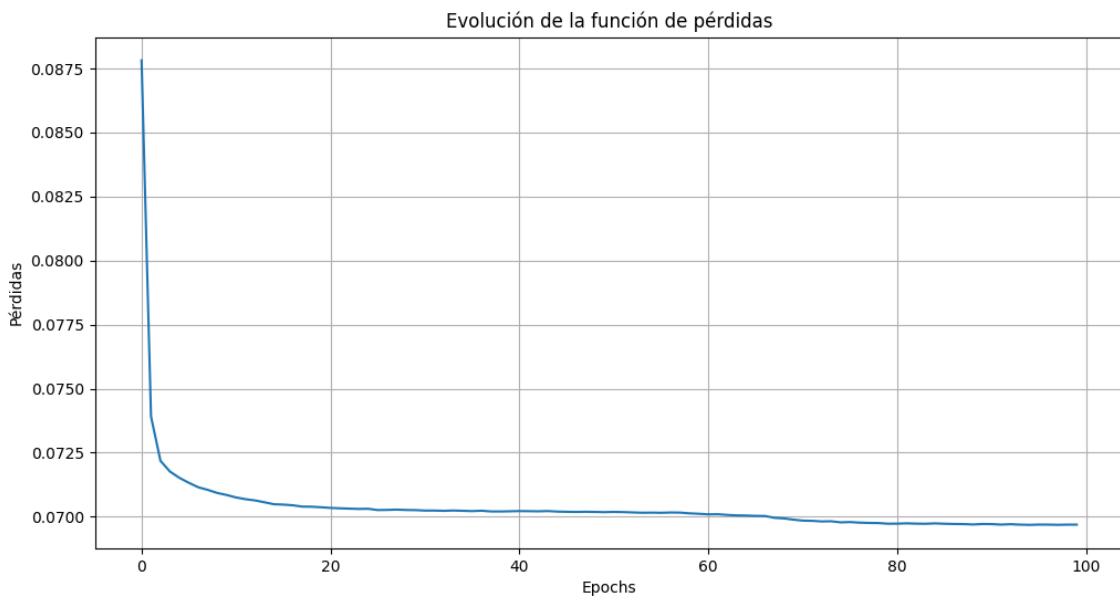


Figura 4.13: Representación del valor de la función de pérdidas en función de las iteraciones para el modelo de ANN

<i>Matriz de confusión</i>	<i>TN</i>	<i>FP</i>	<i>FN</i>	<i>TP</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
<i>sklearn</i>	376535	609	7706	1358	97,84	69,03	14,98	24,66
<i>keras</i>	376674	470	7842	1222	97,84	72,17	13,49	22,74

Tabla 4.10: Resultados de aplicación de la matriz de confusión a las ANNs

<i>Métodos propios de evaluación</i>	<i>Accuracy</i>	<i>Loss</i>
<i>sklearn (score)</i>	97,83	-
<i>keras (evaluate)</i>	97,85	0,0690

Tabla 4.11: Resultados de aplicación de los métodos *score* y *evaluate* a las ANNs

5. Conclusiones y trabajo futuro

En este Capítulo final, se establecerán las conclusiones principales a las que se ha llegado con la realización de este proyecto de investigación y se valorará el cumplimiento de los objetivos expuestos inicialmente en el Capítulo 1. De la misma forma, se explorarán las posibles vías de trabajo futuro que se presentan como opciones de continuación de este TFM.

5.1. Conclusiones del TFM

En este TFM, se han desarrollado diferentes modelos de ML y DL para la detección y predicción precisa de errores en SGs durante el proceso de distribución energética que lleva a cabo el algoritmo DEN2NE.

Este proceso se ha estructurado en seis etapas, comenzando por el estudio del estado del arte y la investigación de las posibles fuentes de datos de implementaciones SGs reales a emplear. Por consiguiente, ha sido necesario evaluar el volumen de información útil de los datos reales, provenientes de la implementación de una SG en la ciudad de Funchal, Madeira. Cabe destacar que se ha aplicado un procesamiento exhaustivo, además de que ha sido necesario incorporar un análisis en profundidad de dos herramientas (*Global Solar Atlas* y *PVWatts*) para simular los datos de producción. Este paso de tratamiento de datos ha sido fundamental y, aunque ha supuesto una mayor extensión en el tiempo de lo estimado inicialmente, se ha logrado un conjunto de datos reales coherente, riguroso y simplificado que abarque muestras tomadas por hora durante un intervalo de un año.

No obstante, después de este procesamiento, se han requerido dos pasos adicionales para extraer el conjunto de datos final sobre el que desarrollar y entrenar los modelos. Por un lado, se ha aplicado la herramienta BRITE para el planteamiento y generación de topologías de red, lo que ha derivado en la creación de 180 en total. Estas topologías se han importado a un simulador propietario realizado en python y han supuesto la base de las simulaciones de distribución energética en DEN2NE.

Por otro lado, también ha sido necesario modelar y realizar una serie de modificaciones en la estructura y en el funcionamiento del algoritmo. Este proceso ha sido imprescindible para permitir la importación al mismo de los datos reales, ejecutar múltiples simulaciones de las topologías generadas en el paso anterior y exportar el conjunto de datos final con las instancias ya etiquetadas. En base a la configuración de los parámetros de DEN2NE se ha establecido la posibilidad de realizar hasta 47.304.000 simulaciones, teniendo en cuenta todo el rango temporal de medidas, que como se ha comentado es de un año. En el caso de este TFM, para obtener suficientes muestras útiles y variadas, se han seleccionado 12 instantes temporales sobre los que trabajar. Esta selección de datos se ha aplicado a la entrada de DEN2NE, derivando a su salida en un conjunto de datos final compuesto por un total de 1.931.040 instancias etiquetadas.

El siguiente paso ha consistido en el desarrollo y entrenamiento de los diferentes modelos de ML y DL, a partir de los patrones de error etiquetados y exportados del algoritmo anteriormente. En este caso, se han planteado, diseñado y probado tres técnicas diferentes: RF, SVM y ANNs, siendo la última desarrollada en dos librerías diferentes (*sklearn* y *keras*), a modo comparativo. El proceso de optimización de los hiperparámetros característicos de cada una de ellas y la aplicación de diferentes métodos de reducción de las dimensiones del conjunto de datos, en el caso del RF y SVM, han derivado en el desarrollo de múltiples opciones a probar.

De la misma forma, la evaluación del desempeño de los modelos ha sido fundamental para realizar un análisis exhaustivo de su rendimiento y para definir el caso óptimo que permita detectar los errores de forma precisa. A modo de sintetizar los resultados finales del desarrollo de este TFM, y de observar las métricas obtenidas, se representa la Tabla 5.1. A partir de la misma, se determina de forma concluyente que el modelo que proporciona el mejor rendimiento en términos globales es el RF con la aplicación del método RFECV.

Finalmente, tomando en consideración todo el trabajo realizado, se puede expresar como conclusión que este TFM ha supuesto un gran reto por su complejidad a nivel técnico y por su extensión temporal requerida. Los resultados obtenidos del desarrollo de este TFM han sido fruto de realizar un trabajo amplio y riguroso, tanto en el estudio, análisis y procesamiento de los datos, como en la creación de modelos de detección y predicción de errores con una alta efectividad. Por ello, este trabajo contribuye significativamente a la mejora del funcionamiento del algoritmo DEN2NE y a la eficiencia y la fiabilidad de las SGs en el contexto de la distribución energética. Adicionalmente, la adaptabilidad de los modelos se extiende al contexto de implementación en entornos de redes de dispositivos IoT, lo que incrementa la importancia de este proyecto y destaca su notable flexibilidad.

		<i>Matriz de confusión</i>			<i>K-Fold Cross Validation</i> <i>*score() / **evaluate()</i>		
		<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>	<i>Accuracy</i>	<i>Standard Deviation / (*, **)Loss</i>
<i>Random Forest</i> [25 estimadores, criterio de entropía]	<i>Sim aplicar</i>	99,29	94,40	74,27	83,16	99,30	0,02
	<i>RFFECV</i>	99,44	94,28	81,05	87,17	99,47	0,02
	<i>kbest (n=5)</i>	97,79	65,97	12,55	21,08	97,80	0,02
	<i>kbest (n=8)</i>	97,74	53,95	27,37	36,32	97,79	0,01
	<i>PCA (n=2)</i>	97,55	12,52	00,67	01,27	97,35	0,01
	<i>PCA (n=4)</i>	98,06	81,66	22,60	35,41	98,04	0,00
<i>SVM</i> [C=1, kernel=rbf]	<i>Sim aplicar</i>	97,23	89,32	06,95	12,83	97,79	0,01
	<i>kbest (n=5)</i>	97,11	65,25	12,29	20,73	97,80	0,01
	<i>kbest (n=8)</i>	97,05	71,19	09,30	16,46	97,79	0,01
	<i>PCA (n=2)</i>	97,61	-	0	-	97,76	0,00
<i>ANN</i> [(5, 5), relu, sgd]	<i>sklearn</i>	97,84	69,03	14,98	24,66	97,83	*
	<i>keras</i>	97,84	72,17	13,49	22,74	97,85	**0,0690

Tabla 5.1: Síntesis de resultados obtenidos del desarrollo de los modelos de ML y DL

5.2. Líneas de trabajo futuro

Tras exponer el alcance logrado de los objetivos del presente TFM, se introducen las posibles vías de trabajo futuro para la continuación de este proyecto de investigación.

- Ampliar el estudio del comportamiento del algoritmo DEN2NE y de los patrones de error a partir de otros conjuntos de datos reales. De esta forma, se podría realizar diversas evaluaciones, a partir de otras fuentes de implementaciones reales de SGs en ubicaciones diferentes.
- Desarrollar técnicas adicionales de ML y DL y evaluar el rendimiento de los nuevos modelos. En este caso, se propone plantear el empleo de Máquinas de Refuerzo de Gradiente (del inglés *Gradient Boosting Machines* (GBM)), como *XGBoost*, ya que permiten manejar grandes cantidades de datos. Sería de interés comprobar si se mejoran los resultados de precisión obtenidos en este proyecto.
- Profundizar el empleo de los modelos para estudiar y predecir otros parámetros relativos al comportamiento de la distribución energética que simula DEN2NE. Se podría plantear un estudio de la predicción del balance de cargas, lo que seguiría mejorando las funcionalidades del algoritmo y proporcionaría una visión más amplia de su funcionamiento.
- Simular en DEN2NE otros instantes temporales del conjunto de datos. En este proyecto se han seleccionado 12 instantes temporales para comprender un día por cada mes del año, suponiendo datos suficientes para entrenar los modelos desarrollados de forma rigurosa. Como línea futura, se podrían seleccionar y probar otras muestras temporales.
- Extender el desarrollo realizado en este TFM al contexto de redes de dispositivos IoT. Por su carácter flexible, sería posible implementar la detección y predicción de errores. En este entorno, se podría basar la fuente de error en las posibles sobrecargas computacionales de los dispositivos.

En definitiva, las líneas de trabajo futuro presentadas proporcionan diversas oportunidades de ampliación del proyecto de investigación realizado, lo que indica el gran alcance que permitiría su continuación.

Bibliografía

- [1] “¿Qué son las Smart Grids o redes inteligentes? | Repsol,” <https://www.repsol.com/es/energia-futuro/tecnologia-innovacion/smart-grids/index.cshtml>, [Accessed 27-10-2023].
- [2] “El impacto de las Smart Grids y la evolución de la Inteligencia Artificial en las redes eléctricas,” <https://es.linkedin.com/pulse/el-impacto-de-las-smart-grids-y-la-evolucion-de-la-inteligencia-artificial-en-las-redes-electricas>, 2023, [Accessed 27-10-2023].
- [3] D. Carrascal, E. Rojas, J. A. Carral, I. Martinez-Yelmo, and J. Alvarez-Horcajo, “Topology-aware scalable resource management in multi-hop dense networks [Unpublished],” 2023.
- [4] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: an approach to universal topology generation,” in *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2001, pp. 346–353.
- [5] NetIS, “BRITE: Boston University Representative Internet Topology Generator,” <https://github.com/NETSERV-UAH/BRITE>, [Accessed 04-04-2024].
- [6] T. Vijayapriya, “Smart grid: An overview,” *Smart Grid and Renewable Energy*, vol. 02, pp. 305–311, 01 2011.
- [7] I. S. Grid, “IEEE Smart Grid 2015 Annual Report,” <https://smartgrid.ieee.org/about-ieee-smart-grid>, 2015, [Accessed 10-01-2024].
- [8] U. D. of Energy, “2009 Smart Grid System Report (July 2009),” <https://www.energy.gov/oe/articles/2009-smart-grid-system-report-july-2009>, 2009, [Accessed 14-01-2024].
- [9] “Smartgrids, transacciones de energía basadas en blockchain,” <https://iotfutura.com/2018/12/smargrid-basada-en-iot-y-blockchain/>, 2018, [Accessed 10-01-2024].

- [10] H.-P. c. Roy Pratt, "Resisting Temptation: the Seven Sins of Smart Grid," <https://www.power-grid.com/news/resisting-temptation-the-seven-sins-of-smart-grid/#gref>, 2010, [Accessed 10-01-2024].
- [11] "Referente mundial en 'smart grids,'" <https://www.iberdrola.com/conocenos/nuestra-actividad/smart-grids>, [Accessed 26-02-2024].
- [12] P. Gyasi, "Smart grids and demand response: Conceptual review," 09 2019.
- [13] W. Lou, S. Zhu, J. Ding, T. Zhu, M. Wang, L. Sun, F. Zhong, and X. Yang, "Transactive demand-response framework for high renewable penetrated multi-energy prosumer aggregators in the context of a smart grid," *Applied Sciences*, vol. 13, p. 10083, 09 2023.
- [14] Iberdrola, "DSO, ¿cómo transformar la gestión de redes hacia un modelo más inteligente?" <https://www.iberdrola.com/innovacion/operadores-de-sistemas-de-distribucion>, [Accessed 14-01-2024].
- [15] J. Iria and F. Soares, "An energy-as-a-service business model for aggregators of prosumers," *Applied Energy*, vol. 347, p. 121487, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306261923008516>
- [16] G. Sun, S. Shen, S. Chen, Y. Zhou, and Z. Wei, "Bidding strategy for a prosumer aggregator with stochastic renewable energy production in energy and reserve markets," *Renewable Energy*, vol. 191, pp. 278–290, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0960148122005274>
- [17] Z. M. Bakare Mutiu Shola, Abdulkarim Abubakar, "A comprehensive overview on demand side energy management towards smart grids: challenges, solutions, and future direction," *Energy Informatics*, vol. 6, 03 2023.
- [18] L. Tao and Y. Gao, "Real-time pricing for smart grid with distributed energy and storage: A noncooperative game method considering spatially and temporally coupled constraints," *International Journal of Electrical Power & Energy Systems*, vol. 115, p. 105487, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0142061519317867>
- [19] A.-H. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *Smart Grid, IEEE Transactions on*, vol. 1, pp. 320 – 331, 01 2011.
-

- [20] “Overview of CHP (Combined Heat & Power),” <https://www.a1energy.net/blogs/post/Overview-of-CHP-Combined-Heat-Power>, 2021, [Accessed 12-01-2024].
- [21] P. Wolfrum, M. Kautz, and J. Schäfer, “Smart operation of chp units,” *IFAC Proceedings Volumes*, vol. 45, no. 21, pp. 61–66, 2012, 8th Power Plant and Power System Control Symposium. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1474667016319450>
- [22] J. Moore and B. Shabani, “A critical study of stationary energy storage policies in australia in an international context: The role of hydrogen and battery technologies,” *Energies*, vol. 9, p. 674, 08 2016.
- [23] K. Sayed and H. Gabbar, “Chapter 18 - scada and smart energy grid control automation,” in *Smart Energy Grid Engineering*, H. A. Gabbar, Ed. Academic Press, 2017, pp. 481–514. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128053430000188>
- [24] B. Shively, “What Is a Phasor Measurement Unit and How Does it Make the Grid More Reliable?” https://www.enerdynamics.com/Energy-Currents_Blog/What-Is-a-Phasor-Measurement-Unit-and-How-Does-it-Make-the-Grid-More-Reliable.aspx, [Accessed 01-03-2024].
- [25] B. T. Rolf Grünbaum, Mojtaba Noroozian, “FACTS, poderosos sistemas para una transmisión flexible de la energía,” https://library.e.abb.com/public/89318066821a283ec1256fda003b4d40/5_1999s.pdf, 1999, [Accessed 04-01-2024].
- [26] M. Donsion, J. Guemes, and J. Rodriguez, “Power quality. benefits of utilizing facts devices in electrical power systems,” in *2007 7th International Symposium on Electromagnetic Compatibility and Electromagnetic Ecology*, 2007, pp. 26–29.
- [27] M. A. Masoum and E. F. Fuchs, “Chapter 9 - the roles of filters in power systems and unified power quality conditioners,” in *Power Quality in Power Systems and Electrical Machines (Second Edition)*, second edition ed., M. A. Masoum and E. F. Fuchs, Eds. Boston: Academic Press, 2015, pp. 779–886. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128007822000099>
- [28] E. Csanyi, “Using HVDC Technology For Transmitting Electricity,” <https://electrical-engineering-portal.com/using-hvdc-technology-for-transmitting-electricity>, 2012, [Accessed 18-01-2024].
-

- [29] S. Cuthrell, “Offshore Wind Needs Major HVDC Transmission Expansion,” <https://eepower.com/news/offshore-wind-needs-major-hvdc-transmission-expansion/>, 2023, [Accessed 18-01-2024].
- [30] N. Mostafa, H. S. M. Ramadan, and O. Elfarouk, “Renewable energy management in smart grids by using big data analytics and machine learning,” *Machine Learning with Applications*, vol. 9, p. 100363, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827022000597>
- [31] D. Wang, X. Guan, T. Liu, Y. Gu, Y. Sun, and Y. Liu, “A survey on bad data injection attack in smart grid,” 12 2013, pp. 1–6.
- [32] J. Jiménez, “Suplantación de ARP: qué es y cómo afecta a nuestra red,” <https://www.redeszone.net/tutoriales/redes-cable/ataques-arp-spoofing-evitar/>, 2023, [Accessed 15-01-2024].
- [33] “Descubrimiento seguro de vecino de IPv6,” <https://www.juniper.net/documentation/mx/es/software/junos/neighbor-discovery/topics/topic-map/ipv6-secure-neighbor>, 2021, [Accessed 15-01-2024].
- [34] H. Shahinzadeh, J. Moradi, G. B. Gharehpetian, H. Nafisi, and M. Abedi, “Internet of energy (ioe) in smart power systems,” 02 2019, pp. 627–636.
- [35] S. G. Info, “Más de 1.300 millones de euros para reforzar y digitalizar la red eléctrica en España,” <https://www.smartgridsinfo.es/2024/06/04/mas-1300-millones-euros-para-reforzar-digitalizar-red-electrica-espana>, [Accessed 05-06-2024].
- [36] “¿Qué es la Internet de las cosas? Definición y explicación,” <https://www.kaspersky.es/resource-center/definitions/what-is-iot>, [Accessed 24-01-2024].
- [37] “¿Qué es IoT (Internet de las cosas)?” <https://aws.amazon.com/es/what-is/iot/>, [Accessed 24-01-2024].
- [38] “IOT VS M2M,” <https://ausum.cloud/iot-vs-m2m-cuales-son-las-diferencias/>, [Accessed 05-06-2024].
- [39] S. Violino, S. Figorilli, C. Costa, and F. Pallottino, “Internet of beer: A review on smart technologies from mash to pint,” *Foods*, vol. 9, p. 950, 07 2020.
- [40] NetIS, “DEN2NE - Distributed ENergy ENvironments and NEtworks,” <https://github.com/NETSERV-UAH/den2ne-Alg>, [Accessed 10-11-2023].
-

- [41] “Las 7 V del Big data: Características más importantes,” <https://www.iic.uam.es/innovacion/big-data-caracteristicas-mas-importantes-7-v/>, 2019, [Accessed 12-01-2024].
- [42] Telefónica, “Las 5 V del Big Data. ¿Cómo pueden beneficiar a las empresas?” <https://www.telefonica.com/es/sala-comunicacion/blog/5-v-big-data-beneficios/>, 2020, [Accessed 13-01-2024].
- [43] B. Bhattacharai, S. Paudyal, Y. Luo, M. Mohanpurkar, K. Cheung, R. Tonkoski, R. Hovsepian, K. Myers, R. Zhang, P. Zhao, M. Manic, S. Zhang, and X. Zhang, “Big data analytics in smart grids: State-of-the-art, challenges, opportunities, and future directions,” *IET Smart Grid*, vol. 2, 03 2019.
- [44] “Diferencias entre la nube y la virtualización,” <https://www.redhat.com/es/topics/cloud-computing/cloud-vs-virtualization>, 2023, [Accessed 13-01-2024].
- [45] “Qué es la Inteligencia Artificial,” <https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr>, 2023, [Accessed 23-02-2024].
- [46] “Qué es la Inteligencia Artificial,” <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-artificial-intelligence>, 2023, [Accessed 23-02-2024].
- [47] O. Sanseviero, “Ai en 3 minutos: Tipos de machine learning,” <https://medium.com/ai-learners/ai-en-3-minutos-tipos-de-machine-learning-945b708ac78>, [Accessed 23-02-2024].
- [48] “Difference Between Artificial Intelligence vs Machine Learning vs Deep Learning,” <https://www.geeksforgeeks.org/difference-between-artificial-intelligence-vs-machine-learning-vs-deep-learning/>, 2023, [Accessed 23-02-2024].
- [49] “Tipos de aprendizaje en machine learning: supervisado y no supervisado,” <https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje>, 2021, [Accessed 26-02-2024].
- [50] “Los conceptos de machine learning y deep learning en la industria,” <https://www.interempresas.net/MetalMecanica/Articulos/347471-Los-conceptos-de-Machine-Learning-y-Deep-Learning-en-la-industria.html>, 2021, [Accessed 26-02-2024].
- [51] R. Yehoshua, “Random forests,” <https://medium.com/@roiyeoh/random-forests-98892261dc49>, 2023, [Accessed 28-02-2024].

- [52] S. Ronaghan, “The mathematics of decision trees, random forest and feature importance in scikit-learn and spark,” <https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2018>, [Accessed 28-02-2024].
- [53] “Decision trees,” <https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation>, [Accessed 29-02-2024].
- [54] “Unlocking the ideas behind of svm (support vector machine),” <https://medium.com/@sachinsoni600517/unlocking-the-ideas-behind-of-svm-support-vector-machine-1db47b025376>, 2023, [Accessed 26-02-2024].
- [55] J. A. Rodrigo, “Máquinas de vector soporte (svm) con python,” <https://cienciadedatos.net/documentos/py24-svm-python>, 2020, [Accessed 26-02-2024].
- [56] “Support vector machines explained with python examples,” <https://towardsdatascience.com/support-vector-machines-explained-with-python-examples-cb65e8172c85>, 2020, [Accessed 26-02-2024].
- [57] “Introducción a support vector machine (svm),” <https://es.mathworks.com/discovery/support-vector-machine.html>, [Accessed 26-02-2024].
- [58] C. Liu, “Svm hyperparameter tuning using gridsearchcv,” <https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/>, 2020, [Accessed 27-02-2024].
- [59] S. Saxena, “The gaussian rbf kernel in non linear svm,” <https://medium.com/@suvigya2001/the-gaussian-rbf-kernel-in-non-linear-svm-2fb1c822aae0>, 2020, [Accessed 27-02-2024].
- [60] “Dlops: Mlops for deep learning,” <https://valohai.com/blog/dlops/>, [Accessed 26-02-2024].
- [61] “¿qué son las redes neuronales?” <https://www.ibm.com/es-es/topics/neural-networks>, [Accessed 02-03-2024].
- [62] F. Bre, J. Gimenez, and V. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks,” *Energy and Buildings*, vol. 158, 11 2017.
- [63] D. Calvo, “Función de activación, redes neuronales,” <https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/>, 2018, [Accessed 02-03-2024].

- [64] B. Pandey, “How do neural networks make decisions? a look at activation functions,” <https://www.goglides.dev/bkpandey/how-do-neural-networks-make-decisions-a-look-at-activation-functions-141e>, 2023, [Accessed 02-03-2024].
- [65] “Router Waxman,” https://www.cs.bu.edu/brite/user_manual/node13.html, [Accessed 05-06-2023].
- [66] “Router BarabasiAlbert (Barabasi-Albert model),” https://www.cs.bu.edu/brite/user_manual/node14.html, [Accessed 05-06-2023].
- [67] E. Zegura, K. Calvert, and S. Bhattacharjee, “How to model an internetwork,” *Proceedings - IEEE INFOCOM*, vol. 2, 02 2002.
- [68] J. Figueras, “¿Conoces NILM? Aplica Inteligencia Artificial en la gestión energética,” <https://www.linkedin.com/pulse/conoces-nilm-aplica-inteligencia-artificial-en-la-gesti%C3%B3n-figueras/>, 2018, [Accessed 23-01-2024].
- [69] A. Monacchi, D. Egarter, W. Elmenreich, S. D'Alessandro, and A. M. Tonello, “Greend: An energy consumption dataset of households in italy and austria,” in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2014, pp. 511–516.
- [70] Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, “Building power consumption datasets: Survey, taxonomy and future directions,” *Energy and Buildings*, vol. 227, p. 110404, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S037877882030815X>
- [71] S. Makonin, B. Ellert, I. V. Bajić, and F. Popowich, “Electricity, water, and natural gas consumption of a residential house in canada from 2012 to 2014,” *Scientific data*, vol. 3, no. 1, pp. 1–12, 2016.
- [72] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, “Blued: A fully labeled public dataset for event-based non-intrusive load monitoring research,” *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, pp. 1–5, 01 2012.
- [73] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, “The eco data set and the performance of non-intrusive load monitoring algorithms,” in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*,

- ser. BuildSys '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 80–89. [Online]. Available: <https://doi.org/10.1145/2674061.2674064>
- [74] S. Makonin, “Hue: The hourly usage of energy dataset for buildings in british columbia,” *Data in Brief*, vol. 23, p. 103744, 03 2019.
- [75] N. Batra, M. Gulati, A. Singh, and M. Srivastava, “It’s different: Insights into home energy consumption in india,” 11 2013.
- [76] J. Kolter and M. Johnson, “Redd: A public data set for energy disaggregation research,” *Artif. Intell.*, vol. 25, 01 2011.
- [77] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, “Smart*: An open data set and tools for enabling research in sustainable homes,” *Proc. SustKDD.*, 01 2012.
- [78] L. Pereira, F. Quintal, R. Gonçalves, and N. J. Nunes, “Sustdata: A public dataset for ict4s electric energy research,” *Proceedings of the 2014 conference ICT for Sustainability*, 2014.
- [79] J. Kelly and W. Knottenbelt, “The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes,” *Scientific Data*, vol. 2, no. 150007, 2015.
- [80] D. P. B. Renaux, F. Pottker, H. C. Ancelmo, A. E. Lazzaretti, C. R. E. Lima, R. R. Linhares, E. Oroski, L. d. S. Nolasco, L. T. Lima, B. M. Mulinari, J. R. L. d. Silva, J. S. Omori, and R. B. d. Santos, “A dataset for non-intrusive load monitoring: Design and implementation,” *Energies*, vol. 13, no. 20, 2020. [Online]. Available: <https://www.mdpi.com/1996-1073/13/20/5371>
- [81] K. Hühne, “Nilm datasets,” <https://github.com/k-nut/nilm-datasets/blob/master/nilm-datasets.csv>, [Accessed 23-01-2024].
- [82] “Clasificación climática de köppen,” https://es.wikipedia.org/wiki/Clasificaci%C3%B3n_clim%C3%A1tica_de_K%C3%B6ppen, [Accessed 30-01-2024].
- [83] “Global solar atlas info,” <https://globalsolaratlas.info/>, [Accessed 30-01-2024].
- [84] “Energy data info,” <https://energydata.info/>, [Accessed 30-01-2024].
- [85] “Epsg - georepository,” https://epsg.org/crs_4326/WGS-84.html, [Accessed 30-01-2024].

- [86] “Global solar atlas 2.0 technical report,” <https://solargis.com/maps-and-gis-data/download/world>, [Accessed 30-01-2024].
 - [87] “Fundamentos energía solar y fotovoltaica,” <https://www.sfe-solar.com/noticias/articulos/energia-fotovoltaica-radiacion-geometria-recorrido-optico-irradiancia-y-hsp/>, [Accessed 31-01-2024].
 - [88] “Pvwatts calculator,” <https://pvwatts.nrel.gov/pvwatts.php>, [Accessed 02-02-2024].
 - [89] “Poa beam,” <https://pvpmc.sandia.gov/modeling-guide/1-weather-design-inputs/plane-of-array-poa-irradiance/calculating-poa-irradiance/poa-beam/>, [Accessed 03-02-2024].
 - [90] A. Prakash, “Threading vs multiprocessing in python: A comprehensive guide,” <https://medium.com/@arjunprakash027/threading-vs-multiprocessing-in-python-a-comprehensive-guide-cae3ce0ca6c1>, [Accessed 06-02-2024].
 - [91] “sklearn.ensemble.randomforestclassifier,” <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>, [Accessed 11-03-2024].
 - [92] “Importancia de la permutación vs la importancia de las características del bosque aleatorio (mdi),” https://qu4nt.github.io/sklearn-doc-es/auto_examples/inspection/plot_permutation_importance.html, [Accessed 09-03-2024].
 - [93] “Tuning the hyper-parameters of an estimator,” https://scikit-learn.org/stable/modules/grid_search.html#grid-search, [Accessed 09-03-2024].
 - [94] “sklearn.model_selection.gridsearchcv,” https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV, [Accessed 10-03-2024].
 - [95] “sklearn.feature_selection.rfecv,” http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html, [Accessed 10-03-2024].
 - [96] “sklearn.feature_selection.selectkbest,” https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest, [Accessed 10-03-2024].
 - [97] “sklearn.decomposition.pca,” <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA>, [Accessed 11-03-2024].
-

- [98] “sklearn.tree.export_graphviz,” https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html, [Accessed 14-03-2024].
- [99] “Metrics and scoring: quantifying the quality of predictions,” https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix, [Accessed 11-03-2024].
- [100] “Graphviz,” <http://www.webgraphviz.com/>, [Accessed 14-03-2024].
- [101] “Data science and machine learning bootcamp with r,” <https://www.udemy.com/course/data-science-and-machine-learning-bootcamp-with-r/>, [Accessed 13-03-2024].
- [102] “Cross-validation: evaluating estimator performance,” https://scikit-learn.org/stable/modules/cross_validation.html#computing-cross-validated-metrics, [Accessed 12-03-2024].
- [103] “sklearn.svm.SVC,” <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>, [Accessed 17-03-2024].
- [104] “Playground tensorflow,” <https://playground.tensorflow.org/>, [Accessed 25-03-2024].
- [105] “sklearn.neural_network.MLPClassifier,” https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, [Accessed 22-03-2024].
- [106] “tf.keras.Sequential,” https://www.tensorflow.org/api_docs/python/tf/keras/Sequential, [Accessed 26-03-2024].
- [107] “Earlystopping,” https://keras.io/api/callbacks/early_stopping/, [Accessed 26-03-2024].

Lista de Acrónimos y Abreviaturas

AMI	<i>Advanced Metering Infrastructure.</i>
ANN	<i>Artificial Neural Network.</i>
API	<i>Application Programming Interface.</i>
ARP	<i>Address Resolution Protocol.</i>
BRITE	<i>Boston University Representative Internet Topology Generator.</i>
CHP	<i>Combined Heat and Power.</i>
DEN2NE	<i>Distributed ENergy ENvironments and Networks.</i>
DIF	<i>Diffuse Irradiance Fraction.</i>
DL	<i>Deep Learning.</i>
DNI	<i>Direct Normal Irradiation.</i>
DoS	<i>Denial of Service.</i>
DSM	<i>Demand Side Management.</i>
DSO	<i>Distribution System Operator.</i>
EPSG	<i>European Petroleum Survey Group.</i>
ESMAP	<i>Energy Sector Management Assistance Program.</i>
FACTS	<i>Flexible AC Transmission System.</i>
FDI	<i>False Data Injection.</i>
GBM	<i>Gradient Boosting Machines.</i>
GHI	<i>Global Horizontal Irradiation.</i>
GTI	<i>Global Irradiation at Optimum Tilt.</i>
HMAC	<i>Hash Message Authentication Code.</i>
HVAC	<i>High Voltage Alternating Current.</i>
HVDC	<i>High Voltage Direct Current.</i>
IA	<i>Inteligencia Artificial.</i>
IEEE	<i>Institute of Electrical and Electronics Engineers.</i>
IoE	<i>Internet of Energy.</i>
IoT	<i>Internet of Things.</i>
IPFC	<i>Interline Power Flow Controller.</i>

LPWAN	<i>Low Power Wide Area Networks.</i>
M2M	<i>Machine To Machine.</i>
MITM	<i>Man In The Middle.</i>
ML	<i>Machine Learning.</i>
MLP	<i>MultiLayer Perceptron.</i>
NILM	<i>Non-Intrusive Load Monitoring.</i>
NLP	<i>Natural Language Processing.</i>
NREL	<i>National Renewable Energy Laboratory.</i>
P2G	<i>Power-to-Gas.</i>
PAR	<i>Peak-to-Average Ratio.</i>
PCA	<i>Principal Component Analysis.</i>
PMU	<i>Phasor Measurement Unit.</i>
POA	<i>Plane of Array Irradiance.</i>
PVOUT	<i>Photovoltaic Power Potential.</i>
PVT	<i>Photovoltaic Thermal.</i>
RBF	<i>Radial Basis Function.</i>
RF	<i>Random Forest.</i>
RFECV	<i>Recursive Feature Elimination with Cross Validation.</i>
RSA	<i>Rivest, Shamir y Adleman.</i>
RTP	<i>Real Time Pricing.</i>
SCADA	<i>Supervisory Control and Data Acquisition.</i>
SEND	<i>Secure Neighbor Discovery.</i>
SG	<i>Smart Grids.</i>
SG	<i>Smart Grid.</i>
SGD	<i>Stochastic Gradient Descent.</i>
SHA	<i>Secure Hash Algorithm.</i>
SINAIS	<i>Sustainable Interactions with Social Networks, context Awareness and Innovative Services.</i>
SSL	<i>Secure Sockets Layer.</i>
SSSC	<i>Static Synchronous Series Compensator.</i>
STATCOM	<i>Static Synchronous Compensator.</i>
SVC	<i>Static Var Compensator.</i>
SVD	<i>Singular Value Decomposition.</i>
SVM	<i>Support Vector Machine.</i>
TCSC	<i>Thyristor-Controlled Series Capacitor.</i>
TES	<i>Thermal Energy Storages.</i>

TFM	Trabajo Fin de Máster.
TLS	<i>Transport Layer Security Protocol.</i>
TOU	<i>Time-of-Use Pricing.</i>
UAH	<i>Universidad de Alcalá.</i>
UPFC	<i>Unified Power Flow Controller.</i>
VMs	Máquinas Virtuales.

A. Anexo I - Pliego de condiciones

Siguiendo la recomendación oficial de la UAH sobre TFMs, se añade el presente anexo para establecer las condiciones materiales de los diferentes equipos que han sido necesarios para desarrollar el proyecto. Se pretende garantizar que se obtengan los mismos resultados en el caso de querer replicar las diferentes pruebas realizadas en el proyecto.

A.1. Condiciones materiales y equipos

A continuación, se indican todos los equipos empleados en el desarrollo de este TFM y sus especificaciones de *software*.

A.1.1. Especificaciones Máquina A

- Procesador: i7-6700K (8) @ 4.200GHz
- Memoria: 31978MiB
- Gráfica: Intel HD Graphics 530
- Sistema operativo: Ubuntu 20.04.6 LTS x86_64

A.1.2. Especificaciones Máquina B

- Procesador: Intel i9-13900k (32) @ 5.500GHz
- Memoria: 31871MiB
- Gráfica: Intel Device a780
- Sistema operativo: Ubuntu 22.04.3 LTS x86_64

```
arppath@server9:~$ neofetch
  .-/+oossssoot+/-.
   `:+ssssssssssssssssssssss+`:
   -+ssssssssssssssssssssyyssss+-.
   .osssssssssssssssssssdMMMNyssso.
   /ssssssssssshdmnNnmmyNMNMNhssssss/
   +ssssssssssshmydMMMMNMNddddyssssss+.
   /ssssssssshNMMyhyyyyhmNMNMNhssssssy/
   .ssssssssdMMMNhsssssssssshNMMDssssss.
   +sssshhhyNMMyyssssssssssyNMMyssssss+.
   ossyNMNMNyMMhssssssssssssshmmmhssssss
   ossyNMNMNyMMhssssssssssssshmmmhssssss
   +sssshhhyNMMyyssssssssssyNMMyssssss+.
   .ssssssssdMMMNhsssssssssshNMMDssssss.
   /ssssssssshNMMyhyyyyhdNMNMNhssssssy/
   +ssssssssdmydMMMMNMNddddyssssss+.
   /sssssssssshdNNNNmyNMNMNhssssss/
   .osssssssssssssssssdMMNyssso.
   -+ssssssssssssssssyyssss+-.
   `:+ssssssssssssssssss+`:
   .-/+oossssoot+/-.
```

arppath@server9

OS: Ubuntu 20.04.6 LTS x86_64
Host: MS-7978 2.0
Kernel: 5.15.0-101-generic
Uptime: 41 days, 6 hours, 4 mins
Packages: 1802 (dpkg), 9 (snap)
Shell: bash 5.0.17
Resolution: 3840x1080
Theme: Adwaita [GTK3]
Icons: Adwaita [GTK3]
Terminal: /dev/pts/0
CPU: Intel i7-6700K (8) @ 4.200GHz
GPU: Intel HD Graphics 530
Memory: 1235MiB / 31978MiB



Figura A.1: Especificaciones de la máquina A

```
arppath@coruscant:~$ neofetch
  .-/+oossssoot+/-.
   `:+ssssssssssssssssssss+`:
   -+ssssssssssssssssssssyyssss+-.
   .osssssssssssssssssssdMMNyssso.
   /ssssssssssshdmnNnmmyNMNMNhssssss/
   +sssssssssshmydMMMMNMNddddyssssss+.
   /ssssssssshNMMyhyyyyhmNMNMNhssssssy/
   .ssssssssdMMMNhssssssssssshNMMDssssss.
   +sssshhhyNMMyyssssssssssyNMMyssssss+.
   ossyNMNMNyMMhssssssssssssshmmmhssssss
   ossyNMNMNyMMhssssssssssssshmmmhssssss
   +sssshhhyNMMyyssssssssssyNMMyssssss+.
   .ssssssssdMMMNhssssssssssshNMMDssssss.
   /ssssssssshNMMyhyyyyhdNMNMNhssssssy/
   +ssssssssdmydMMMMNMNddddyssssss+.
   /sssssssssshdNNNNmyNMNMNhssssss/
   .osssssssssssssssssdMMNyssso.
   -+ssssssssssssssssyyssss+-.
   `:+ssssssssssssssss+`:
   .-/+oossssoot+/-.
```

arppath@coruscant

OS: Ubuntu 22.04.3 LTS x86_64
Host: MS-7D25 2.0
Kernel: 6.5.0-26-generic
Uptime: 6 days, 1 hour, 28 mins
Packages: 1999 (dpkg), 11 (snap)
Shell: bash 5.1.16
Resolution: 1924x768
Terminal: /dev/pts/4
CPU: 13th Gen Intel i9-13900K (32) @ 5.500GHz
GPU: Intel Device a780
Memory: 3986MiB / 31871MiB



Figura A.2: Especificaciones de la máquina B

A.1.3. Especificaciones Máquina C

- Procesador: Intel(R) Core(TM) 12th Gen i7-1265U CPU @ 1.80Ghz
- Memoria: 15674MiB
- Gráfica: Intel Iris Xe Graphics
- Sistema operativo: Windows 11 Pro

A.2. Virtualización del sistema operativo Linux

Adicionalmente, es preciso indicar que se ha empleado una máquina virtual con un sistema operativo Linux para ejecutar la herramienta BRITE en la máquina C. En este caso, se ha utilizado como plataforma VMWare¹ y una versión del sistema operativo Ubuntu 20.04.1 LTS como se puede visualizar en la Figura A.3.

Device Name	ubuntu >
Memory	1.9 GiB
Processor	AMD® Ryzen 5 3400g with radeon vega graphics × 2
Graphics	SVGA3D; build: RELEASE; LLVM;
Disk Capacity	21.5 GB
<hr/>	
OS Name	Ubuntu 20.04.1 LTS
OS Type	64-bit
GNOME Version	3.36.3
Windowing System	X11
Virtualization	VMware

Figura A.3: Características del sistema operativo

¹<https://www.vmware.com/es/products.html>

B. Anexo II - Presupuesto

Este anexo pretende estimar el presupuesto que sería necesario para llevar el proyecto a cabo. Por ello, se cuantifican las horas efectivas dedicadas al TFM y se establecen los costes, en base a los medios *hardware* y *software* empleados y a la mano de obra.

B.1. Duración del proyecto

Para obtener el número de horas de trabajo por semana del presente proyecto en **promedio** se aporta un diagrama de Gantt en la Figura B.1, en el que se puede apreciar cómo se han distribuido las tareas a lo largo del tiempo. Como aclaración, la duración del proyecto se ha extendido en el tiempo varias semanas más de lo que se había previsto e indicado en el anteproyecto. Principalmente, esto es porque se ha requerido un diseño y procesamiento exhaustivo de los datos para obtener el conjunto final sobre el que desarrollar y entrenar los diferentes modelos. Como se puede ver, en la Figura B.1 se reflejan las etapas críticas del proyecto, que vienen dadas por las de diseño de datos y de desarrollo de los modelos de ML y DL.

Teniendo en cuenta esta distribución temporal, en la Tabla B.1 se estima de forma aproximada el total de horas dedicadas al TFM y, por ende, el número de horas efectivas por semana en **promedio**.

Número de horas totales	Horas efectivas	Horas efectivas por semana
600h	≈ 390-520h	≈ 15-20h

Tabla B.1: Promedio de horas de trabajo

B.2. Costes del proyecto

En cuanto al cálculo de los costes del proyecto, se aplica una diferenciación en términos del *hardware*, del *software* y de la mano de obra que se necesita. De esta forma, se pueden desglosar de forma detallada los costes y, por tanto, aportar de forma aproximada la

cuantía total del proyecto. Esto es importante para garantizar una consideración adecuada de todos los recursos que se requieren para desarrollar de forma exitosa el proyecto.

Por un lado, se pone el foco en los costes asociados al *hardware* (ver Tabla B.2) y al *software* (ver Tabla B.3). En el caso del primero, se requiere evaluar todos los equipos o dispositivos que han sido necesarios para desarrollar el TFM, mientras que para el segundo, es preciso tener en cuenta el gasto anual que suponen las licencias o paquetes de software empleados.

Producto (IVA incluido)	Valor (€)
Ordenador portátil HP EliteBook 640 G9	1296,00
Servidor A	2100,00
Servidor B	1700,00

Tabla B.2: Presupuesto desglosado del hardware para el TFM

Producto (IVA incluido)	Valor (€)
Microsoft Office	300,00

Tabla B.3: Presupuesto desglosado del software para el TFM

Por otro lado, se estiman los costes de mano de obra o, en otros términos, los gastos relacionados con los recursos humanos que se han involucrado en el proyecto. Para ello, se emplea como referencia el salario promedio de un ingeniero junior y se cuantifica el total a partir del número de horas totales dedicadas al TFM.

Una vez desglosados los costes diferenciados, se presenta en la Tabla B.4 el presupuesto total de desarrollo del proyecto. Cabe destacar que todos los cálculos expresados se basan en las referencias proporcionadas y pueden ser ajustados según los requerimientos específicos del proyecto, así como los precios aplicables a cada caso particular.

Descripción (IVA incluido)	Unidades	Coste unitario (€)	Coste total (€)
Material Hardware	1	5096,00	5096,00
Material Software	1	300,00	300,00
Mano de obra	600	15,00	9000,00
Costes fijos (Luz, Internet)	1	300,00	300,00
TOTAL			6596,00€

Tabla B.4: Presupuesto total con IVA

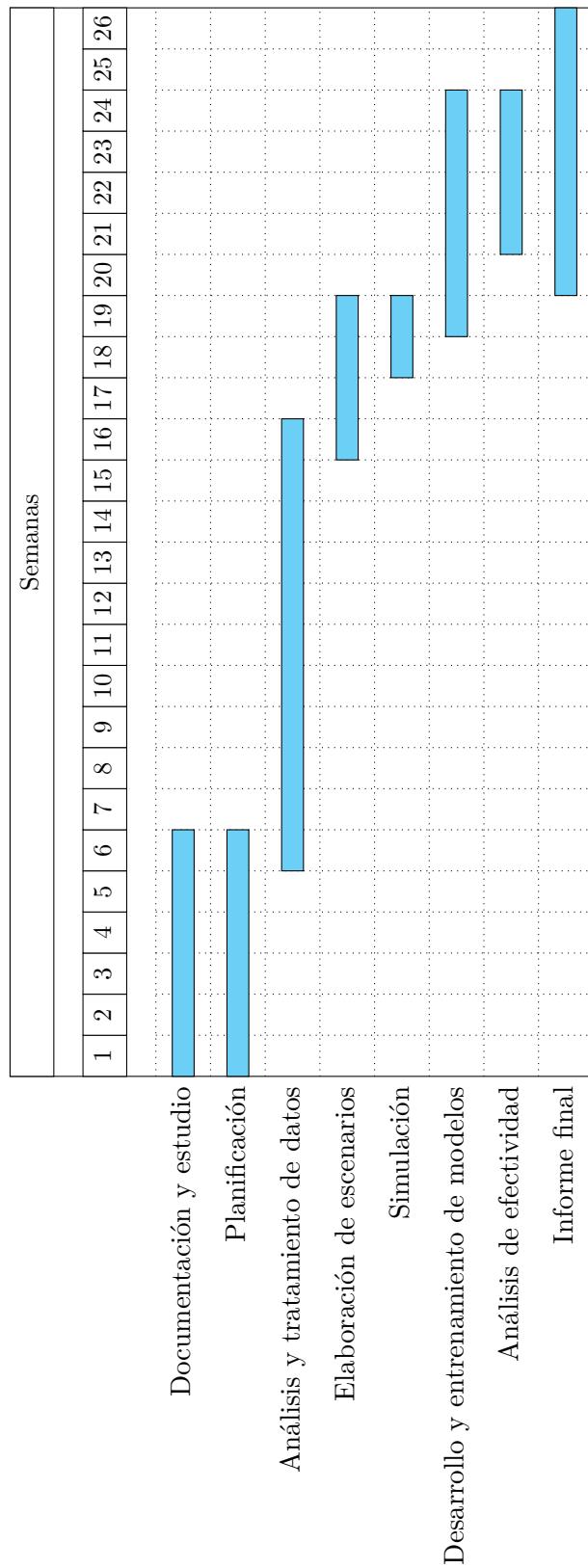


Figura B.1: Diagrama de Gantt del proyecto

C. Anexo III - Manuales de usuario e instalación

A modo de facilitar al lector el acceso y el uso de las herramientas empleadas en este TFM, se dedicará este anexo a la creación de un manual de usuario. Se desarrollarán de forma detallada los procedimientos de instalación y configuración del software empleado, además de exponer su funcionamiento completo. De la misma forma, se incluirá la información necesaria para comprender conceptualmente los scripts que se han empleado con el fin de automatizar y optimizar el desarrollo de los casos prácticos.

C.1. Herramienta BRITE

El objetivo de incluir esta sección es exponer al lector la información de instalación y configuración de la plataforma de generación de topologías de red BRITE, descrita previamente en la Sección 2.6.1.

C.1.1. Compilación e instalación de BRITE

Para proceder a instalar la herramienta, se accede al repositorio del equipo de investigación NetIS de la UAH¹ y se descarga y se descomprime el fichero .zip con todos los contenidos necesarios. Después, se realiza un *make* desde el directorio raíz de la herramienta. Se pueden producir errores por la falta de paquetes *g++* o *openjdk++*, que se soluciona con los comandos dados en el Código C.1.

Código C.1: Instalación de paquetes

```
1 sudo apt install g++
2 sudo apt install openjdk-11-jdk
```

¹<https://github.com/NETSERV-UAH/BRITE>

C.1.2. Ejecución de BRITE

Se puede ejecutar BRITE mediante la interfaz gráfica (GUI) (ver Código C.2) o por comandos. En el caso de los comandos se da la posibilidad de utilizar los ejecutables programados en C++ o en Java (ver Código C.3), obteniéndose los mismos resultados a partir de una misma entrada.

Código C.2: Ejecución de la interfaz gráfica de BRITE

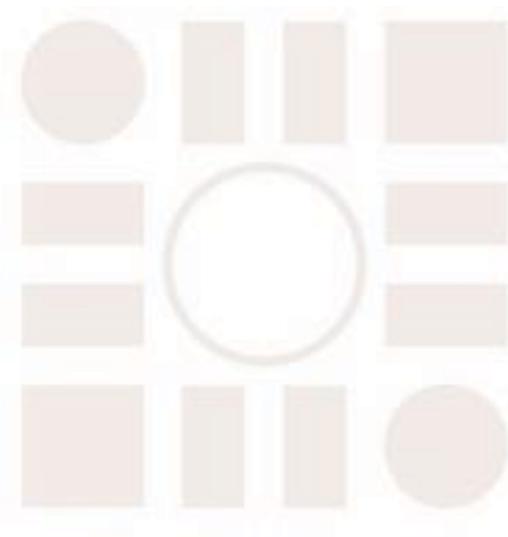
```
1 ./britegui &
```

Código C.3: Ejecución de BRITE por la línea de comandos

```
1 $bin/brite <archivo de entrada .conf> <archivo de salida .brite> <seed_file>
2 $C++/cppgen <archivo de entrada .conf> <archivo de salida .brite> <seed_file>
```

Como se especifica en el Código C.3, se requiere un archivo de configuración de entrada (.conf), en el cual se determinen los parámetros de la topología, y otro de salida, especificado en formato .brite. Como se ha expuesto en el Capítulo referente al Estado del Arte (ver Sección 2.6.1.3), para facilitar el uso de BRITE se aportan en el repositorio una serie de ficheros de Python y scripts dedicados a la automatización del proceso de creación de los ficheros de configuración y del tratamiento de la salida.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR

