UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL MODELADO DE PANELES SOLARES"

TRABAJO FIN DE GRADO

Julio - 2021

AUTOR: Aarón Manzano Navarro

DIRECTOR/ES: Vicente Galiano Ibarra

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1
INTRODUCCIÓN

En el presente trabajo se va a realizar una predicción del comportamiento de las placas solares mediante la Inteligencia Artificial (2.1), las cuales variarán dependiendo de la ciudad y la tecnología que posea.

A modo introductorio, se podría decir que la investigación consta de tres partes claramente diferenciadas. En primer lugar, se seleccionarán una serie de datos relacionados con las características propias de la placa solar, de los que nos interesan unos parámetros determinados. Estos datos serán obtenidos gracias a la fuente de datos de NREL (Laboratorio Nacional De Energías Renovables). A continuación, se les realizará un preprocesamiento para transformarlos y reducir el valor de los errores encontrados para, en último lugar, quedarnos con los datos más útiles que nos permitan entrenar eficientemente la red neuronal (2.7). Dicha red se encargará de obtener las predicciones más reales posibles.

Por lo tanto, sería conveniente explicar aspectos importantes que se necesitarán tener en cuenta antes de comenzar con el desarrollo de la investigación.

En el apartado resultados y discusión se verá cómo el primer paso es cargar los datos mencionados, que estarán almacenados en un archivo. Esto lo obtendremos gracias a la fuente de datos de NREL (National Renewable Energy Laboratory). Además, se tendrá en cuenta que los datos que nos interesan son de parámetros concretos, facilitados gracias al método o algoritmo Two Step Linear Least Square (TSLLS) que, a su vez, obtiene el valor de los cinco parámetros del modelo de único diodo (SDM). Pero ¿qué son el modelo SDM y el método TSLLS?

1.1 NREL

Laboratorio Nacional De Energías Renovables o en inglés National Renewable Energy Laboratory es el laboratorio principal de EE. UU para la investigación y el desarrollo (I + D) de energía renovable.

A través de este se ha podido modelar mediante el método TSLLS su repositorio público de curvas I-V, conformado por 1025599 curvas I-V de 22 módulos fotovoltaicos de hasta ocho tecnologías diferentes. A lo largo de un año, se han ido recogiendo medidas cada cinco minutos en tres lugares climáticamente diferentes, como son Cocoa, Eugene y Golden.

En la siguiente ilustración vemos como efectivamente las tres ciudades están ubicadas en tres zonas distintas de EE.UU.

Ilustración 1: Mapa EE.UU

El hecho de que las mediciones estén en tres zonas distintas está pensado para poder contrastar datos. Para una investigación donde lo que tiene mucho peso es la temperatura y la irradiancia, no será lo mismo que durante el año en el que se han tomado las mediciones en una zona tengamos de media una temperatura de 17º y en otra de 26º.

Vamos a ver los gráficos de temperatura que tienen durante un año cada una de estas ciudades.

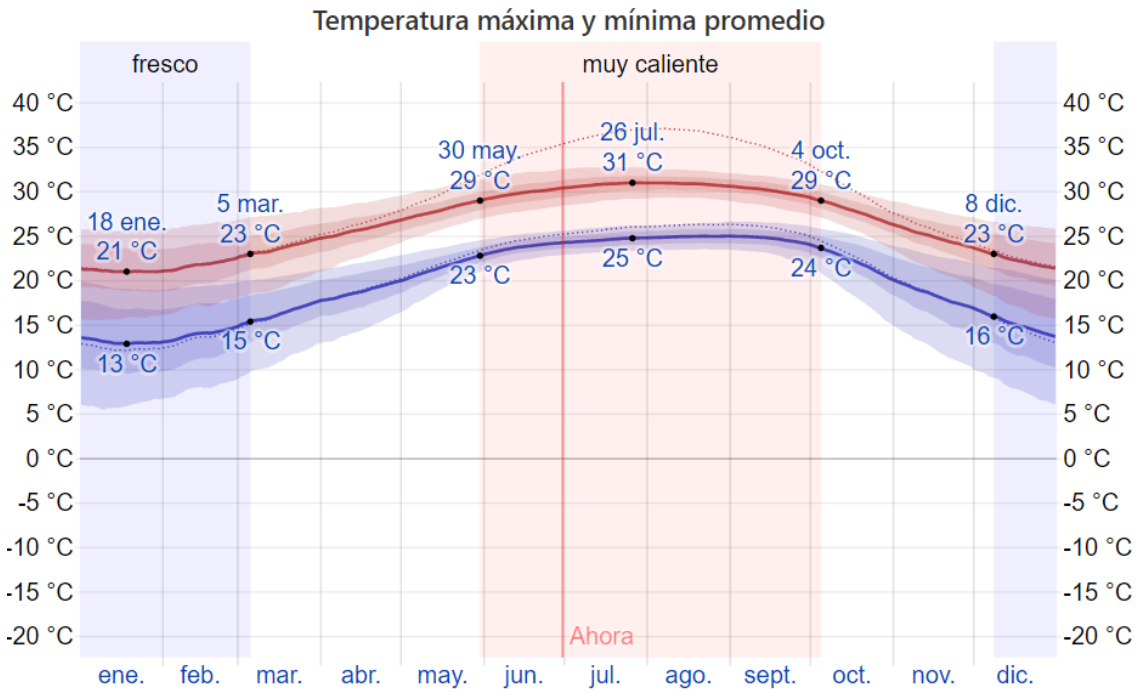

Ilustración 2: Temperatura media anual Cocoa.

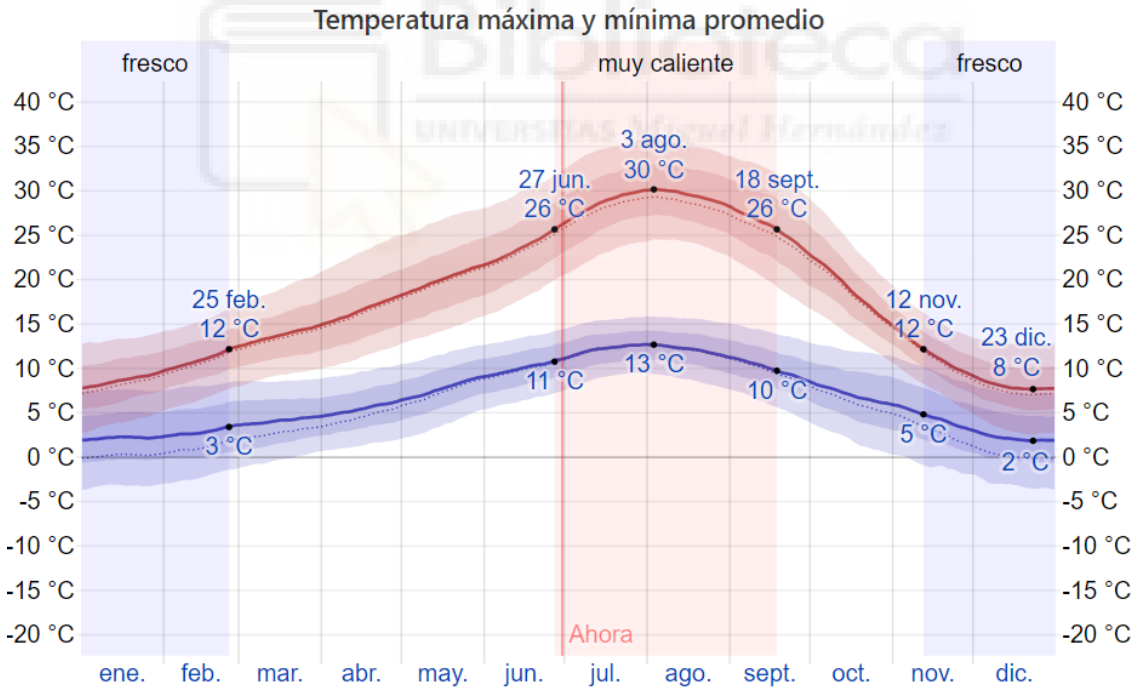

Ilustración 3: Temperatura media anual Eugene.

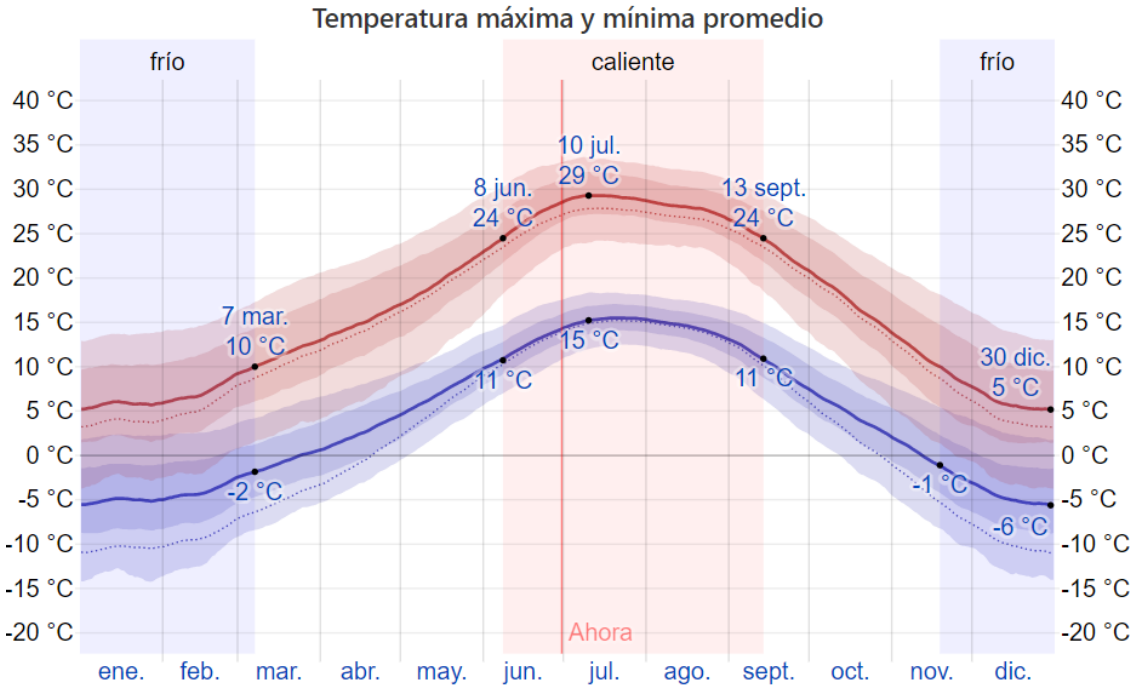

Ilustración 4: Temperatura media anual Golden.

Como vemos en las gráficas anteriores Cocoa es la que durante el año tiene una temperatura más elevada, le sigue Eugene con unas temperaturas algo inferiores durante todo el año. Por último, tenemos la ciudad de Golden que sin duda es la que tiene menor temperatura en todo el año. Por lo tanto, en Cocoa vamos a tener una irrandiancia mayor durante todo el año.

Por último, vamos a ver los módulos fotovoltaicos desplegados en cada una de estas ciudades [1].

Ilustración 5: Equipamiento fotovoltaico Cocoa.

Ilustración 6: Equipamiento fotovoltaico Eugene.

Ilustración 7: Equipamiento fotovoltaico Golden.

1.2 MODELO DE ÚNICO DIODO

Este modelo es uno de los más simples, consta de una fuente de corriente en paralelo a un diodo como veremos en la imagen inferior, donde la salida de dicha fuente es directamente proporcional a la luz que incide sobre la celda.

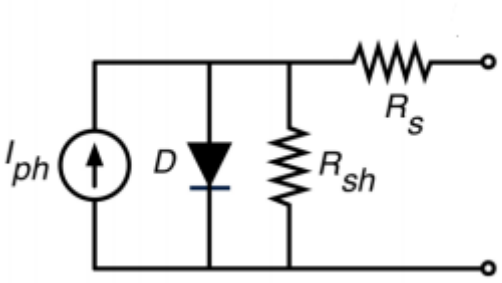

Ilustración 8: Celda placa solar modelo único diodo

Este modelo requiere de tres parámetros para caracterizar la curva I-V, uno es la Intensidad, otro el voltaje del circuito y por último el factor de idealidad del diodo. Vemos que contiene una resistencia en serie (R_s), muy útil para mejorar el comportamiento cuando esta se somete a variaciones ambientales, especialmente a bajas tensiones. Por otro lado, vemos otra resistencia llamada $R_{sh,}$ esta nos sirve de derivación adicional [2].

La ilustración anterior representa lo que sería una sola celda en una placa solar, pero para que la fórmula que se va a representar a continuación tenga sentido es necesario ver el diagrama completo de una placa solar. Esta estará compuesta por una serie de celdas en paralelo como vemos en la siguiente ilustración:

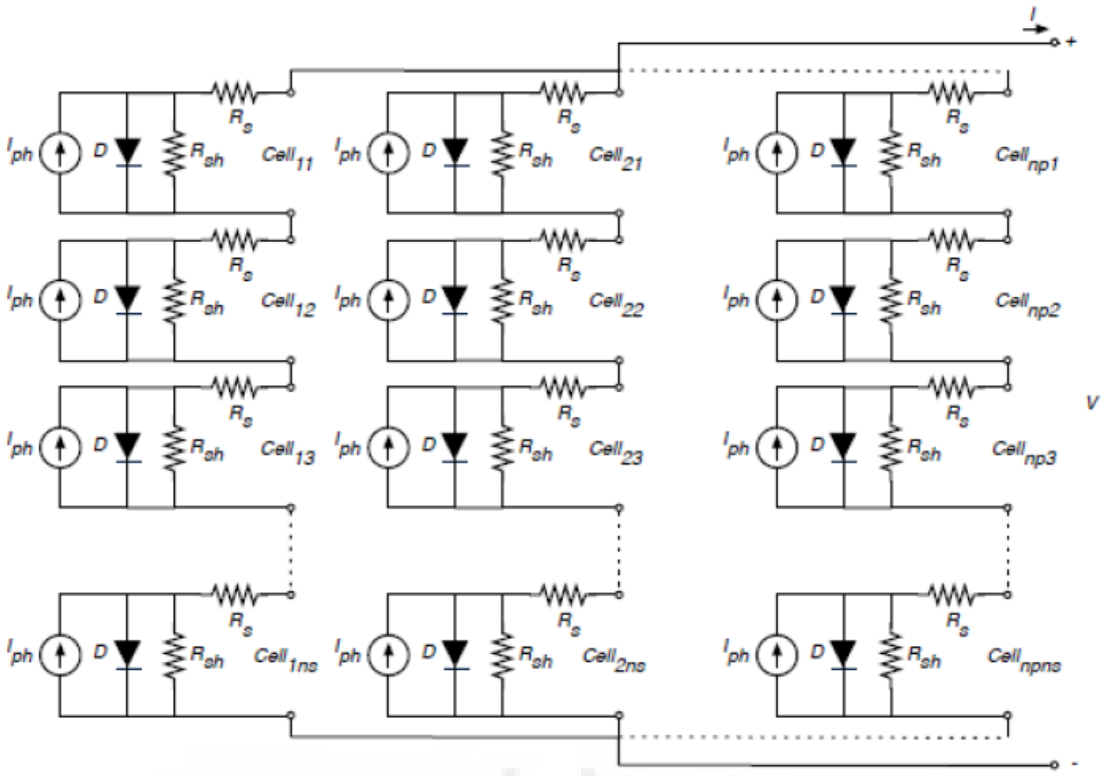

Ilustración 9: Conjunto de celdas de una placa solar

Utilizando el modelo de único diodo, bajo condiciones de trabajo determinadas, se puede relacionar la intensidad y el voltaje mediante la siguiente expresión:

$$I = n_p I_{ph} - n_p I_{sat}\left(e^{\frac{\frac{V}{n_s}+\frac{IR_s}{n_p}}{nV_T}} - 1\right) - n_p \frac{\frac{V}{n_s}+\frac{IR_s}{n_p}}{R_{sh}}$$

Ilustración 10: Fórmula modelo único diodo

donde:

- R_s: resistencia en serie de una celda / panel
- R_{sh}: resistencia en paralelo asociada a cada célula
- I_{ph}: fotocorriente de una celda / panel
- I_{sat}: corriente de saturación de un diodo de una celda / panel
- n: factor de idealidad del diodo.

Una vez identificados los parámetros que intervienen en el modelo y obtener los valores de I-V, mediante su correspondiente medición, se necesitan calcular cuáles son los valores de estos cinco parámetros. Para ello, se hará uso del método TSLLS comentado anteriormente, puesto que la fórmula anterior es compleja y analíticamente es imposible de despejar.

1.3 TSLLS

El método Two step linear least square (método de mínimos cuadrados lineales en dos pasos) también conocido por sus siglas TSLLS, es capaz de trabajar a ciegas con cualquier tipo de curva I-V. Por tanto, como no necesita ninguna presuposición tampoco es necesario conocer información de ningún parámetro.

Cabe destacar, que los resultados obtenidos por este método en una primera etapa tienen el mismo orden de precisión que los mejores métodos documentados en el campo de la extracción de parámetros, pero para más inri en una segunda etapa se obtiene la mejor precisión documentada hasta el momento.

Para la extracción de los parámetros del modelo se han sugerido diversas metodologías, las cuales pueden dividirse en tres grupos: las que tratan de extraer los parámetros a través de la ficha técnica del fabricante (datasheet), las que intentan obtener los parámetros con una colección de puntos experimentales de corriente-tensión (I-V) y, por último, las que utilizan tanto el datasheet como los datos experimentales.

Entre los métodos comentados anteriormente, hay algunos que extraen los parámetros como funciones de la irradiancia y la temperatura, lo que permite que, una vez extraídos los parámetros, se puedan extrapolar para diferentes condiciones de trabajo.

En cuanto a la obtención de los datos requeridos en el trabajo, en particular los valores de los parámetros se adquieren a partir de un conjunto de puntos experimentales de corriente-tensión (I-V). Sin embargo, dichos valores no proporcionan ningún significado físico de los parámetros, por lo que se debe ejecutar cada vez que las condiciones ambientales cambien.

A modo de conclusión, este método es capaz de identificar los cinco parámetros del modelo de diodo único (SDM), haciendo uso de las propiedades geométricas específicas de la curva I-V para extraer directamente los parámetros a partir de un conjunto de puntos I-V [3].

1.4 OBJETIVOS

1.4.1 OBJETIVOS PRINCIPALES

Como se ha comentado, tomaremos una serie de datos que serán obtenidos gracias a la fuente de NREL (National Renewable Energy Laboratory). Este nos permitirá obtener las medidas de las condiciones ambientales (temperatura e irradiancia) y mediante el método TSLLS se obtendrán los cinco parámetros para cada una de estas condiciones ambientales. Por último, necesitamos los parámetros del fabricante de la placa solar, que estarán proporcionados mediante su hoja técnica (datasheet).

Nos planteamos en este momento, crear y entrenar una red neuronal que permita predecir los datos de salida. Por lo tanto, el objetivo es que a posteriori podamos poner el panel solar en diferentes puntos de funcionamiento. Lo que nos permitirá tener en cada momento del día el punto de máxima potencia (MPPT) y así sacarle la mayor eficiencia al panel solar.

1.4.2 OBJETIVOS PERSONALES

Los objetivos personales que me he propuesto alcanzar son varios. El primero es aprender un nuevo lenguaje de programación, Python. Con nuevo no me refiero a que sea moderno, sino que a lo largo de mis años cursando Ingeniería informática no he tenido la suerte de estudiarlo. La principal causa de este objetivo es que es un lenguaje que está al alza y puede abrirme muchas puertas para el futuro. Este aprendizaje incluye adquirir todo lo que lo envuelve, desde la nueva sintaxis, hasta las librerías estándar más interesantes que pueda llegar a permitirme desarrollar código lo más rápido posible.

Mi segundo objetivo es conocer alguno de los campos de la Inteligencia Artificial. Como en el caso anterior, campos como Machine Learning o aprendizaje automático están en auge y a mí, personalmente siempre me ha llamado la atención. De la misma manera, no he tenido la oportunidad de verlo durante mis

años de carrera y gracias a esta propuesta de TFG voy a intentar lograr este objetivo y conocer esto más a fondo.

CAPÍTULO 2 MATERIAL Y MÉTODOS

En este segundo apartado, se van a describir cuáles son los métodos y materiales que han sido utilizados en la realización del trabajo. Para ello, debemos conocer las definiciones de los conceptos necesarios para el desarrollo del trabajo realizado.

2.1 ¿QUÉ ES LA INTELIGENCIA ARTIFICIAL?

Para conocer qué es la inteligencia artificial debemos conocer primero la definición de inteligencia. La Real Academia de la Lengua Española define inteligencia como "*Potencia intelectual: facultad de conocer, entender o comprender*", sabiendo esto podemos un símil, pero con máquinas en vez de con humanos.

La definición dada con anterioridad podría ser un tanto ambigua, por lo que haremos referencia a la definición que realizó **Marvin Minsky**, uno de los pioneros de la IA, que decía lo siguiente: "*La Inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia*".

Un ejemplo podría ser el ajedrez, sería impensable que un ordenador tuviera que evaluar todas y cada una de las jugadas posibles, por lo que, en vez de esto, se le otorga un conocimiento en el proceso de búsqueda para que encuentre la mejor jugada en forma predefinidas o procedimientos de evaluación inteligentes [4].

A continuación, veremos otras definiciones importantes, como son el Machine Learning y Deep Learning para así comprender los métodos y herramientas que se han llevado a cabo en este trabajo.

2.2 HISTORIA DE MACHINE LEARNING

Antes de hablar de qué es el Machine Learning, vamos a ver un poco de historia para conocer cómo surgió este método.

En el año **1943** el matemático **Walter Pitts** y el neurofisiólogo **Warren McCulloch** dieron a conocer su trabajo que estaba orientado a lo que hoy conocemos como Inteligencia Artificial. Estos proponían analizar el cerebro como un organismo computacional para que las computadoras funcionaran igual o mejor que nuestra red neuronal [5].

En **1950**, **Alan Mathison Turing**, científico informático y matemático, entre otras, destacó por su interés en *la relación entre el pensamiento humano y los procesos lógicos de las computadoras* [6]. Turing fue el encargado de crear el conocido "Test de Turing" para así ser capaz de medir la inteligencia de un ordenador al tratar de mantener una conversación, cuya finalidad era la de imitar el comportamiento humano.

Fue el profesor e informático teórico **Arthur Samuel** quien, en el año 1952, dio a conocer el primer programa de cómputo capaz de aprender. Lo más interesante de este programa es que es capaz de aprender, haciéndolo cada vez mejor, obteniendo todas las opciones posibles.

Poco años después, en el **1956**, **Marvin Minsky**, **John McCarthy** y otro grupo de profesionales en medio de una conferencia científica en Darthmouth dan el nombre de "*Artificial Intelligence"*. Poco después, el psicólogo Norte Americano **Frank Rosenblatt** fue el creador de perceptrón, primera red neuronal artificial.

Después de esto, la Inteligencia Artificial quedó apartada por el gran costo que suponía. Sin embargo, en el año 1979 aparecieron un grupo de estudiantes de Ingeniería de la Universidad de Stanford que crearon un robot conocido como "Sanford Car", capaz de moverse autónomamente por una habitación evitando obstáculos. [5]

El proyecto anteriormente comentado no hubiera sido posible sin el algoritmo **Nearest Neighbor**, basado en instancia de tipo supervisado de Machine Learning. Este algoritmo puede utilizarse para clasificar nuevas muestras o directamente para predecir, aunque principalmente se utiliza para clasificar valores buscando los puntos de datos más próximos aprendidos en la etapa de entrenamiento y realizando una hipótesis de nuevos puntos basados en esa clasificación [7].

En **1981**, modelos de software como el de Gerald Dejong, que trabaja bajo su concepto "Explanation-based Learning" (EBL), utilizaban el aprendizaje automático compuesto por cuatro variables: un dominio, un espacio de hipótesis, ejemplos de entrenamiento y criterios de operatividad.

Continuando con la historia de la IA, en **1985** el profesor teórico **Terry Sejknowski** creó "NetTalk", capaz de aprender la pronunciación de palabras a niveles escolares. Cabe destacar que dicho autor fue uno de los miembros de la Academia Nacional de Ciencia de su país.

Llegados a este punto de la historia, la inteligencia artificial volvió a tener un estancamiento y se paralizó completamente, ya que pasaron doce años hasta volver a tener actividad. No obstante, en el **1997** nace el conocido *ordenador Deep Blue* de **IBM**, una de las industrias tecnológicas más grandes de la década, dicho ordenador fue capaz de vender al campeón mundial de ajedrez **Gary Kaspárov**.

Casi diez años más tarde, en **2006** vuelve a aparecer el anteriormente comentado **IBM** pero ahora iría de la mano con **Microsoft.** Estos empezarían a expandirse a nivel global con el tan conocido Machine Learning. Y es así como **Microsoft** lanzó en el **2008** una herramienta con un servicio en la nube que permite a sus usuarios almacenar aplicaciones en el centro de procesamiento de **Microsoft** permitiéndoles ser capaces de ejecutar sobre ella, además aportaba un mayor nivel de confidencialidad que el que podían aportar otras herramientas similares en estos momentos.

Pero **IBM** no se iba a quedar atrás cuando en el año **2011** para hacer más revolucionario el Machine Learning, creó un ordenador llamado **Watson**, que se hizo conocido en un concurso televisivo llamado Jeopardy, donde venció a todos

sus oponentes humanos. Las pruebas constaban de responder a una serie de preguntas, de las que Watson solo presentó problemas en algunas de las categorías que no les brindaban las pistas necesarias para que pudiera responder correctamente.

Si nos fijamos en los hechos sucedidos a lo largo del tiempo, nos damos cuenta de que cada vez habían más industrias y empresas interesadas en este campo en auge. Tanto es así que, en **2012** los amigos **Jeff Dean**, que trabajaba en **Google** y **Adrew N**, profesor de la universidad de Stanford, se pusieron codo a codo para dirigir un innovador proyecto de **Google**, llamado **Google Brain**. La finalidad de este proyecto era ni más ni menos que detectar patrones en vídeos e imágenes mediante una red neuronal.

Este tipo de Red Neuronal Profunda (RNP) se consolidó en sistemas de seguridad y a nivel militar. Además, empresas como **Facebook** pudieron desarrollar algoritmos más complejos como DeepFace, capaz de reconocer a una persona.

Compañías como **DeepMind**, dedicada al Deep Learning no pueden quedar en el olvido. Esta en particular creó un algoritmo capaz de jugar a los juegos de Atari como lo haría una persona. Además, como buen sistema de aprendizaje tras unas cuantas horas de juego podía ser capaz de vender a auténticos profesionales. Fue entonces cuando en **2014**, **Google** compró la compañía.

En este punto, en el año **2015**, aparece el nombre de **Amazon** en este campo que lanza su propia plataforma de Machine learning. A su vez, en este mismo año **Microsoft** decide seguir innovando, creando el kit de herramientas para aprendizaje de máquinas distribuidas (Distribuited Machine Learning Toolkit), que permite la distribución eficiente de problemas de Machine learning en múltiples equipos.

Como estamos viendo era una constante lucha, en cuanto a innovación y destreza, lo que no se esperaba es que, en este mismo año, un conjunto de personas donde se encontraban **Elos Musk** y **Sam Altman** fundaron una organización sin ánimo de lucro llamada **Open AI**, cuyo objetivo es promover y desarrollar inteligencia artificial amigable de tal manera que beneficie a la humanidad.

Tras lo mencionado, cabe destacar que el Machine Learning ha sido, y sigue siendo, muy importante en el campo de la informática. Gracias a todas las aportaciones realizadas a lo largos de estos años, hoy en día se pueden ver grandes avances en sectores como educación, medicina, finanzas, construcción y robótica, entre otros. [5]

2. 3 ¿QUÉ ES EL MACHINE LEARNING?

En el apartado anterior hemos hablado de inteligencia artificial. Esto nos hace profundizar en el Machine Learning (Aprendizaje automático) el cual es una subrama del anteriormente mencionado.

Para explicar que es el Machine Learning (ML) nos basaremos en definiciones con fundamento. Para ello, retrocedemos al año 1959 donde Arthut Samuel, pionero en el campo de la inteligencia artificial lo definió como: "*el campo de estudio que proporciona a las máquinas la capacidad de aprender sin ser programadas de forma explícita*".

Una definición más detallada podría ser la de Dan Fagella, CEO de la empresa Emerj que decía: "*Machine Learning es la ciencia que permite que las computadoras aprendan y actúen como lo hacen los humanos, mejorando su aprendizaje a lo largo del tiempo de una forma autónoma, alimentándolas con datos e información en forma de observaciones e interacciones con el mundo real".*

Un sistema de ML se basa en algoritmos que realizan cálculos fundamentados en una cantidad inmensa de datos. Cuantos más datos obtienen, mejor es el rendimiento de esta.

El objetivo es hacer predicciones con los datos facilitados y el Machine Learning permite realizar este trabajo de manera autónoma, sin necesidad de programar constantemente su evolución [8].

2.4 TIPOS DE APRENDIZAJE EN MACHINE LEARNING

Es importante conocer la clasificación de los algoritmos de ML, ya que, dependiendo de las necesidades del problema, el ambiente en el que se van a desenvolver y los factores que afectarán a la toma de decisiones. Podemos tener distintos tipos de algoritmo de aprendizaje.

En este apartado vamos a tratar 3 tipos que son: supervisado, no supervisado y por refuerzo.

2.4.1 APRENDIZAJE SUPERVISADO (SUPERVISED MACHINE LEARNING)

Este tipo de algoritmo se encarga de generar un modelo predictivo basado en los datos de entrada y salida.

Por lo tanto, podemos decir que se utiliza cuando se dispone de un conjunto de entrenamiento con datos previamente etiquetados, es decir, tener un conjunto de muestra donde ya se conoce al grupo, valor o categoría al que pertenecen. De esta forma el algoritmo va aprendiendo a clasificar las muestras de entrada comparando el resultado del modelo.

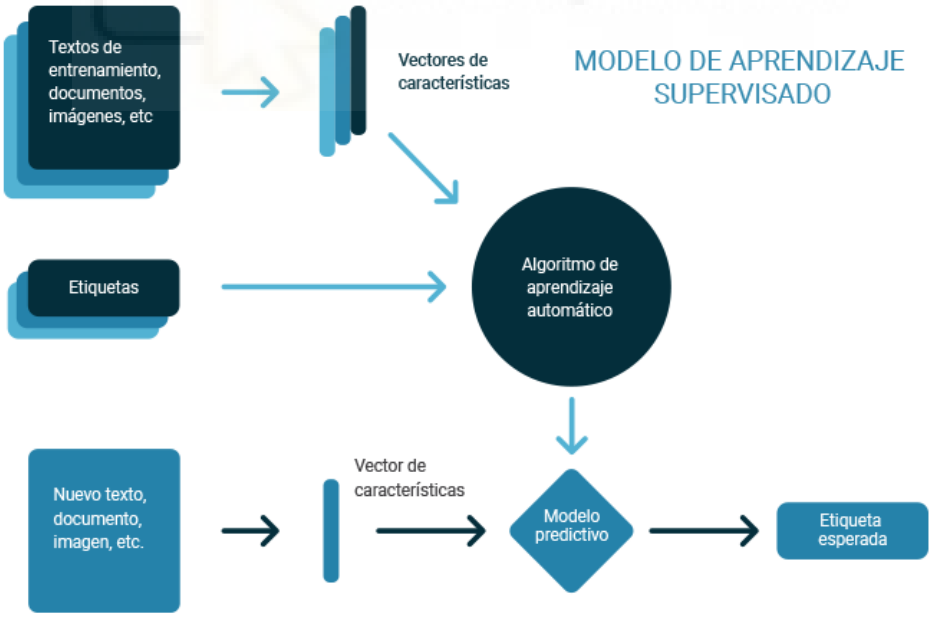

Ilustración 11: Modelo de aprendizaje supervisado

2.4.2 APRENDIZAJE NO SUPERVISADO (UNSUPERVISED MACHINE LEARNING)

En el caso del aprendizaje no supervisado se incluyen conjuntos de datos, sin etiquetar, donde no se conoce previamente la estructura que poseen. A diferencia del supervisado con este se busca obtener información clave o importante sin conocer previamente las variables de salida, explorando la estructura de los datos que no están etiquetados.

En este algoritmo podemos encontrar dos categorías específicas que se conocen como clustering y reducción dimensional. Cuando hablamos de clustering nos referimos a una técnica exploratoria para analizar datos, donde se organiza la información por grupos sin conocer de forma previa la estructura que los compone. La finalidad de esto es la de obtener grupos de datos con características similares.

Por otro lado, tenemos la reducción dimensional donde se utilizan datos de alta complejidad que requieren de mayor capacidad de procesamiento.

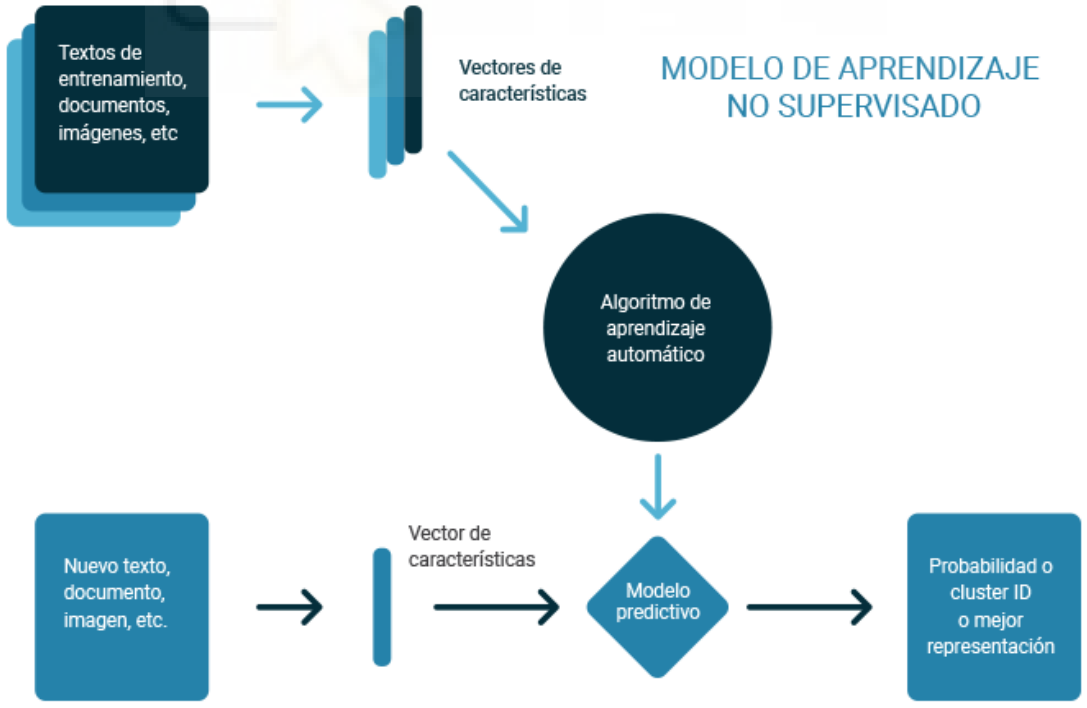

Ilustración 12: Modelo de aprendizaje no supervisado

2.4.3 APRENDIZAJE POR REFUERZO (REINFORCEMENT LEARNING)

Este tipo de aprendizaje forma parte de lo que conocemos como Deep Learning, el cual veremos en el siguiente punto. La finalidad de este algoritmo es la de construir modelos que aumenten el rendimiento tomando como base el resultado o la recompensa que se genera por cada interacción realizada.

Ilustración 13: Modelo aprendizaje por refuerzo

Como conclusión, ningún enfoque es mejor que otro, sino que cada uno tiene sus ventajas. Por lo tanto, el algoritmo a utilizar será el que mejor se adapte al problema [9].

2.5 ¿QUÉ ES DEEP LEARNING?

Una vez desarrollados los puntos anteriores llegamos al quid de la cuestión. Es decir, hemos ido profundizando en la inteligencia artificial hasta llegar a este método.

Debemos saber que el Deep Learning o aprendizaje profundo es un subcampo de Machine Learning, el cual intenta simular una estructura de redes neuronales artificiales como las de los humanos, conectados entre sí en forma de tela de araña.

Mediante un gran conjunto de datos etiquetados hace lo que es natural para los humanos, es decir, aprender con el ejemplo, ya que puede realizar tareas de clasificación directamente desde imágenes, texto o sonido.

Algo importante a destacar es que los modelos que se crean con este método pueden lograr una precisión impresionante, llegando a superar en algunos casos al rendimiento humano [10].

Antes de terminar con las definiciones de Inteligencia Artificial, Machine Learning y Deep Learning, se mostrará a continuación una imagen con las principales diferencias que existen entre ellas:

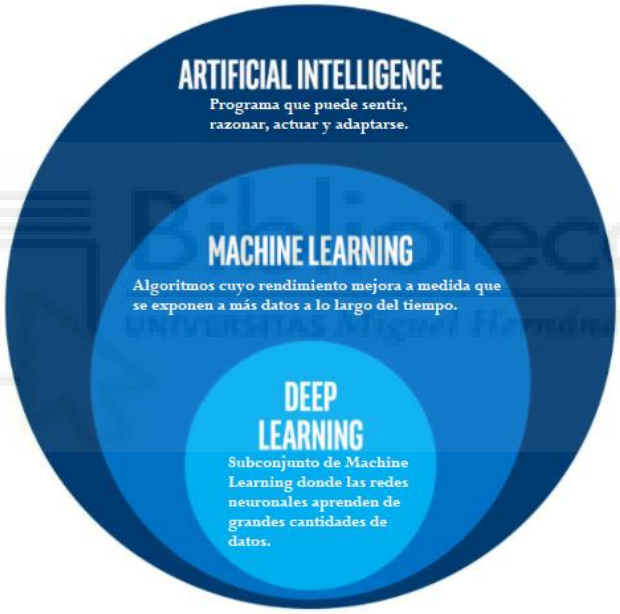

Ilustración 14: Diferencias entre IA, ML y DL

2.6 DIFERENCIAS ENTRE MACHINE LEARNING Y DEEP LEARNING

Anteriormente hemos visto que Deep Learning es un subcampo de Machine Learning, pero eso no quiere decir que uno y otro sean iguales. Para ello vamos a fijarnos en la tabla que se muestra a continuación [11]:

Machine Learning	Deep Learning
Se puede utilizar con cantidades de datos pequeñas.	Requiere gran cantidad de datos de capacitación no etiquetados para llegar a un buen resultado.
No es necesario hardware de alto rendimiento para su uso.	Su exigencia con respecto al hardware es más elevada.
Las características deben de ser identificadas por los usuarios.	Se encarga de crear nuevas características por sí mismo.
Divide las tareas en pedazos pequeños para más tarde combinar los resultados en una conclusión.	Resuelve el problema de principio a fin.
No requiere un tiempo elevado para entrenar.	Requiere más tiempo para entrenar ya que la red neuronal necesita calcular un número significativo de pesos.

Tabla 1: Comparación de Machine Learning con Deep Learning.

2.7 ¿QUÉ ES UNA RED NEURONAL?

En el apartado 2.6 para hablar de Deep Learning hemos tenido que dar una breve pincelada de que son las redes neuronales, ahora vamos a ver esto con un poco más de detalle mostrando también ventajas y desventajas de estas.

Como hemos dicho anteriormente, las redes neuronales tratan de emular ciertas características de los seres humanos, como la capacidad de memorizar. Por lo tanto, son un modelo artificial y simplificado del cerebro humano. Dicho de otra manera, podemos decir que es “un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona”.

Para ponernos en situación, una neurona biológica está estimulada a través de sus entradas (inputs). Cuando esta alcanza un cierto nivel la activa, enviando de esta manera una señal hacia el axón. El pensamiento tiene lugar en el cerebro donde tenemos billones de neuronas interconectadas, por eso mismo es una red de neuronas (neuronal).

Podemos decir que las redes neuronales consisten en unidades de procesamiento que permiten intercambiar datos y/o información para poder aportar mayor conocimiento a esta.

Otra característica importante es que se utilizan para reconocer patrones, que pueden ser de diferentes tipos (imágenes, secuencias de tiempo, etc.).

Por último y como comentábamos con anterioridad, tienen la capacidad de aprender. Sin embargo, esto no es realmente importante ya que lo más interesante es que además mejoran su funcionamiento. Es decir, cuanto más aprenden, mejor funcionamiento desarrolla [12].

2.7.1 VENTAJAS DE LAS REDES NEURONALES

Una vez vista la definición de redes neuronales, vamos a ver cuáles podrían ser sus ventajas y sus desventajas (en el siguiente punto “2.7.2”) [13]. Por lo tanto, podemos decir que las redes neuronales:

- Se entrenan, aprenden y olvidan
- Son robustas y tolerantes a fallas. Gracias a esto si una neurona o varias está teniendo un rendimiento bajo o directamente está fallando no implica que la red neuronal tenga un fallo total.
- Son flexibles, se adaptan perfectamente a nuevos ambientes gracias a su sistema independiente.
- Tiene una velocidad de respuesta incluso mayor que la del cerebro humano.
- Tienen una gran habilidad para asociar, evaluar y reconocer patrones.

2.7.2 DESVENTAJAS DE LAS REDES NEURONALES

Acabamos de ver que las redes neuronales tienen una serie de ventajas, pero esto no puede ser perfecto. A continuación, se enumeran unas desventajas que debemos de tener en cuenta [14]:

- No pueden resolver todos los problemas y tampoco siempre de la mejor manera.

- Su fuerte no es la precisión, lo que se pretende es que sean rápidos y con respuestas buenas ya que estos realizan operaciones sobre problemas de cálculos complejos.
- Las redes neuronales están teniendo un buen comportamiento para predicciones a largo plazo con componentes no lineales, pero no están claras las mejorías que se observan en series cortas y estacionales, como por ejemplo las predicciones de ventas.
- Complejidad de aprendizaje para realizar grandes tareas, si queremos que la red aprenda demasiadas cosas, más complicado será enseñarle.

2.8 LENGUAJE DE PROGRAMACIÓN: PYTHON

Una vez hemos hablado de Inteligencia Artificial y sus subcampos, nos centraremos en profundizar sobre el lenguaje con el que se desarrolla.

El lenguaje de programación que vamos a utilizar, *Python*, es un lenguaje versátil multiplataforma y multiparadigma que se utiliza para Big Data, Inteligencia Artificial (AI), Data Science, entre otros. Por ello, es el idóneo para trabajar con grandes volúmenes de datos, Además, cuenta con una gran biblioteca de recursos, lo que hace que todo tipo de operaciones y funciones matemáticas sean accesibles para todos [15].

2.9 CARACTERÍSTICAS DE PYTHON

Python se califica por ser un lenguaje simple, raudo y por tener una curva de aprendizaje afable y corta. Además, está desarrollado bajo una licencia de código abierto, lo que hace que su uso sea accesible de manera pública y que esté a disposición de cualquier persona para su uso.

- **Interpretado**, es decir, “interpreta” el código del programador, lo traduce y lo ejecuta a la vez.
- Es **Multiparadigma** ya que, admite el uso de varios paradigmas de programación, es decir, varios modelos de desarrollo, de esta manera no requiere que los programadores usen un único estilo de programación, por lo que puede que un programador trabaje mediante programación

orientada a objetos, otro con programación imperativa y un tercero con programación funcional.

- Además, como hemos dicho en el apartado anterior es **multiplataforma**, con lo cual puede ejecutares en diferentes sistemas como Unix, Linux, MacOS y Windows.
- Es de **tipado dinámico**, esto quiere decir que el intérprete asigna a las variables un tipo durante el tiempo de ejecución basado en su valor en ese momento [16].

2. 10 ¿POR QUÉ PYTHON?

Este es un lenguaje que todo el mundo debería de conocer. Como hemos dicho anteriormente, su sintaxis simple y sencilla, el tipado dinámico, su gestor de memoria, la inmensa cantidad de librerías disponibles y el gran poderío que tiene, hace que podamos desarrollar una aplicación de manera rápida. Además, no es necesario poseer gran conocimiento para programar en Python, todo esto es posible con las librerías comentadas anteriormente.

Su sintaxis es tan cercana al lenguaje natural que los programas creados con este parecen pseudocódigo, es otro de los motivos por lo que se usa de inicialización para programar. Ahora bien, este no es adecuado para la programación a bajo nivel o para aplicaciones en las que el rendimiento sea decisivo.

Como dato, habría que destacar una serie de proyectos o programas realizados con Python como son: Netflix, Instagram, Google (Junto con C++ y Java), Yahoo! [17].

2.11 JUPYTER NOTEBOOK

Tras haber hablado de Python es el momento de ver la aplicación web donde se ha tenido que ejecutar el código que se ha ido realizando.

Jupyter Notebook es una interfaz web desarrollada con lenguaje HTML, el cual permite introducir texto, vídeo, imágenes, audio, además de la ejecución de código a través de múltiples lenguajes (entre ellos Python) [18].

Esta aplicación está diseñada generalmente para tener compatibilidad avanzada con Python, Markdown (lenguaje de marcado ligero para maximizar la legibilidad y facilidad de publicación) e incluye la posibilidad de exportar documentos hechos con esta herramienta a otros formatos.

Vamos a ver una serie de características de esta herramienta [19]:

- Fácil instalación.
- Interfaz avanzada para poder combinar código fuente, fórmulas, gráficas y multimedia en un solo documento.
- Fácil accesibilidad desde cualquier lugar sin necesidad de instalar otro servicio (funciona como cliente servidor).
- Como hemos dicho anteriormente, se usa en múltiples lenguajes, pero el lenguaje de programación fundamental en Jupyter Notebook es Python, aunque es compatible con más de 40 lenguajes.
- Se pueden visualizar imágenes, vídeos, LaTeX, entre otros, además es capaz de manipular resultados en tiempo real.
- Los archivos compatibles con esta herramienta podemos visualizarlos, ni que decir tiene que dichos archivos deben de estar alojados en nuestro equipo.
- Es compatible con nbviewer el cual permite portar documentos de Jupyter Notebook a la nube como página estática para así ser visualizada por cualquiera sin necesidad de instalar esta herramienta.

A continuación, vamos a ver una ilustración donde se muestra la interfaz de Jupyter Notebook:

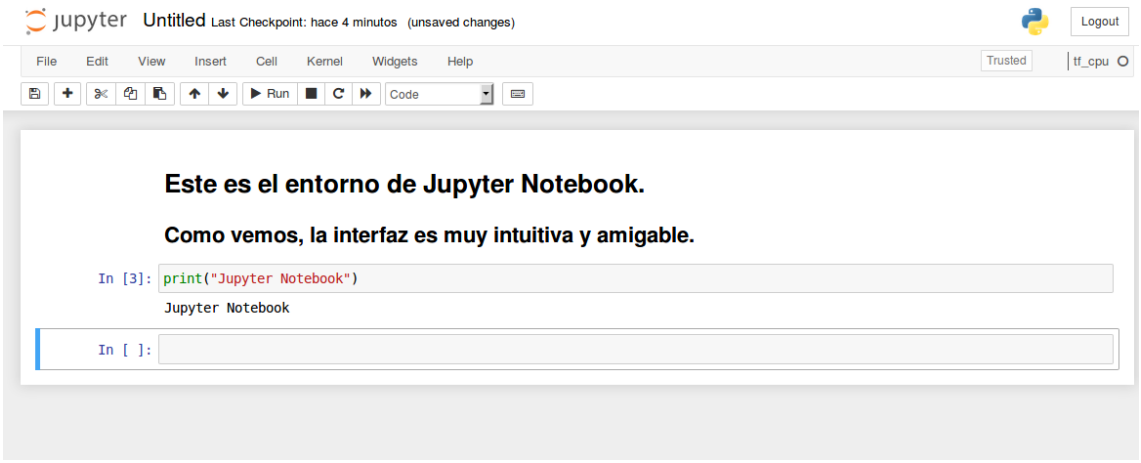

Ilustración 15: Ejemplo Jupyter Notebook

CAPÍTULO 3 RESULTADOS Y DISCUSIÓN

Una vez desarrollados los conceptos que permiten entender el trabajo de investigación efectuado, primero se expondrá la construcción del modelo de nuestra red neuronal. Tras esto, comenzaremos a ver las distintas pruebas realizadas y los resultados obtenidos en cada una de ellas.

3.1 CONSTRUCCIÓN DEL MODELO DE LA RED NEURONAL

Antes de nada, vamos a ver el croquis de cómo será nuestra red neuronal, en el que se detallan los valores de entrada y de salida. De esta forma, podemos apreciar de manera visual la red neuronal utilizada:

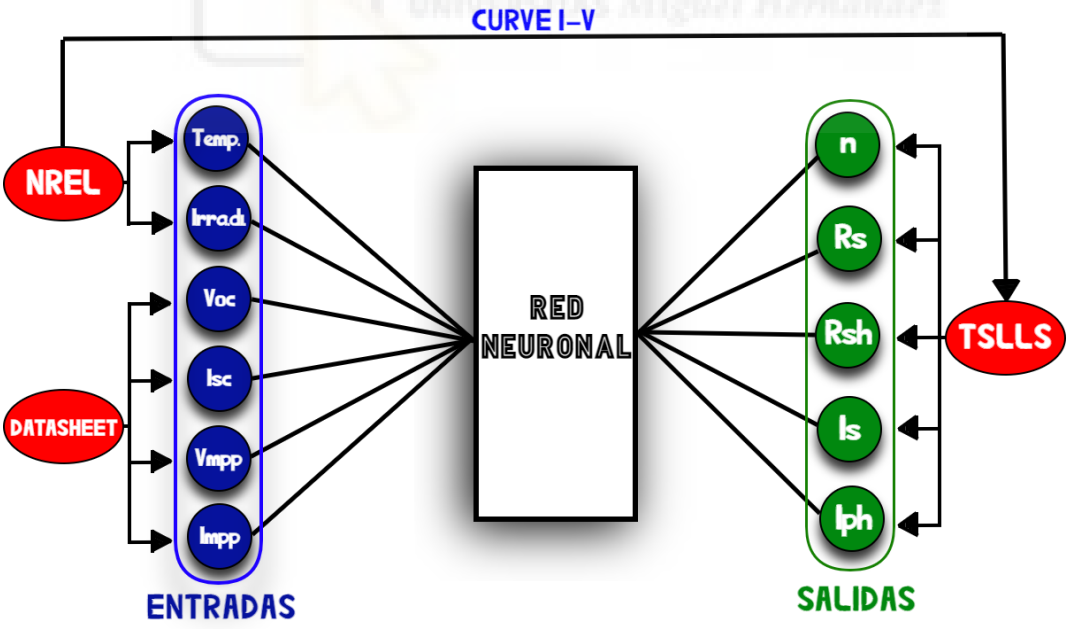

Ilustración 16: Croquis Red Neuronal

Para entrenar la red neuronal utilizaremos como valores de entrada, en un mismo dataframe, las condiciones ambientales (Temperatura e irradiancia) y, los valores del Datasheet (V_{oc}, I_{sc}, V_{mpp} e I_{mpp}).

Además de estos valores, para crear el modelo de la red neuronal también tendremos que introducir los valores de salida que, como hemos visto en la introducción, se obtienen mediante el método TSLLS. Como se puede observar en el esquema anterior, estos parámetros son: n, R_s, R_{sh}, I_{ph} e I_{sat}.

En primer lugar, se ha construido el modelo y se ha entrenado la red neuronal con los datos mencionados. Para construir el modelo, se han normalizado los datos de entrada y de salida, ya que de esta forma las redes neuronales funcionan mejor, si no se realiza esto, los datos de entrada tendrán un efecto adicional sobre la neurona, dando lugar a decisiones incorrectas. Tras dicha normalización, los valores se encuentran en el intervalo [-1,1].

Una vez realizado el paso anterior, se crearán datasets que serán distribuidos de la siguiente forma:

- El 70% de ellos se utilizará para el entrenamiento de la red. Es decir, serán los datos utilizados para entrenar la red neuronal.
- El 15% para realizar el test. Estos datos serán extraídos del conjunto de datos y se utilizarán a posteriori para obtener las predicciones.
- El 15% será utilizado para la validación. Este no está apartado como en el caso del test, digamos que “vive” dentro del conjunto de entrenamiento. Este se utilizará durante las iteraciones que se harán con el conjunto de entrenamiento.

Por último, se realizará la construcción del modelo. Para mostrarlo de manera más gráfica, en lugar de utilizar el código se presenta mediante la siguiente ilustración:

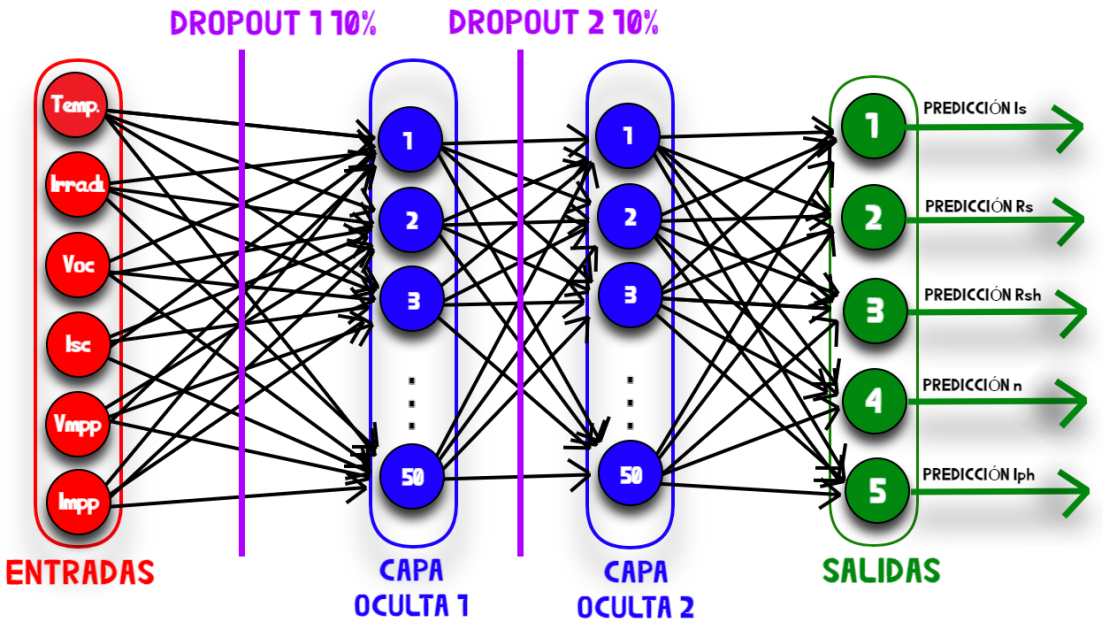

Ilustración 17: Modelo de la Red Neuronal.

Lo primero que vemos es la capa *Input*, digamos que es la puerta de entrada de nuestra red neuronal, compuesta por unas dimensiones concretas para nuestro modelo. Tras esto, el modelo irá encadenando una capa a la siguiente y así aceptando como input la capa anterior.

La siguiente capa que vemos es la capa de *Dropout*, su función principalmente es evitar *overfitting* (Es una de las deficiencias que dificulta la precisión y el rendimiento del modelo). Lo que viene a hacer es tomar un X% de la capa que le entre, y poner a cero esos valores. Cada vez va a ir desconectando valores distintos. Esto sirve para que la red tenga que "buscarse la vida" si en algún momento, por ejemplo, hay datos de entrada que no existan.

La salida de esta capa va directamente conectada a la capa *Dense* la cual es nuestra primera capa oculta. Conformada por 50 neuronas y con una función de activación *Relu.* La función de activación se encarga de devolver una salida a partir de un valor de entrada.

Relu es la función de activación más empleada por sus buenos resultados. Este aplica una transformación no lineal muy simple, activa la neurona solo si el input está por encima de 0. Si en cambio está por debajo, la salida será igual a cero, el valor de salida incrementa de forma lineal con el de entrada. De esta manera,

la función de activación se queda solo con los valores positivos y descarta los negativos proporcionando una activación de cero.

De nuevo, de la salida de la primera capa oculta conectaremos a *Dropout_1* que nos otorgará mayor beneficio con respecto a lo dicho anteriormente de este tipo. Tras este, vuelve a conectarse de *Dropout_1* a *Dense_1* de la misma manera compuesta por 50 neuronas y una función de activación *Relu*.

La salida de este último se conecta a *Concatenate* donde vemos que aparte de *Dropout_1* también le entra la capa *Input*. El hecho de pasarle el input es para darle más información al modelo. Podemos verlo como que le estamos diciendo al modelo: "*Los valores de entrada ya te los doy yo, tú céntrate en aprender otras cosas*".

Por último, tenemos la capa *Output*, este no deja de ser otra capa *Dense*, pero en este caso tiene 5 neuronas y la función de activación es *Sigmoide.* Como vemos en la figura a la capa *Output* le entra todas las capas desarrolladas anteriormente.

La función de activación *Sigmoide* sigue siendo la función por defecto utilizada en las capas de salida. Esta transforma valores en el rango (-inf, +inf) a valores en el rango (0,1).

Llegados a este punto se compila el modelo desarrollado. Es ahora cuando podemos comparar los valores de entrenamiento frente a las predicciones. Es decir, compararemos los valores reales con los que se ha predicho la red neuronal.

En los siguientes apartados vamos a ver estos resultados de diferentes formas, desde observar su comportamiento sin tratar los datos, hasta manipular estos en gran medida.

3.2 TEMPERATURA E IRRADIANCIA FRENTE A PARÁMETROS

Es interesante conocer de qué manera afecta la temperatura e irradiancia de la placa a los valores que se obtienen en los cinco parámetros a analizar. Para este estudio, hemos utilizado los datos proporcionados por la ciudad de Cocoa, concretamente en el panel *aSiMicro_03036*.

- Corriente generada por la irradiación (I_{ph})

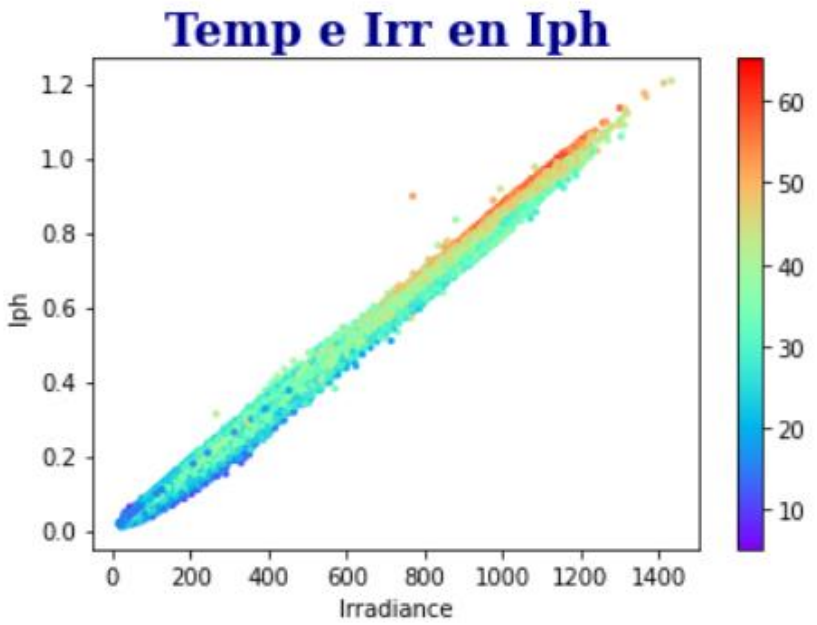

Ilustración 18: Temperatura e Irradiancia en I_{ph}.

Como vemos en el gráfico anterior, la corriente generada por la irradiación es directamente proporcional a la irradiancia. Es decir, a mayor irradiancia, mayor es el valor de I_{ph}. Esto se refleja en la gráfica ya que el valor de I_{ph} crece linealmente a medida que aumenta la irradiancia.

En cuanto a la temperatura, aunque no es una relación directa también se aprecia una correlación. Y es que, los valores más elevados de temperatura precisan de mayores valores de I_{ph}, mientras que los valores más bajos coinciden con las menores temperaturas.

- Corriente de saturación inversa (I_s)

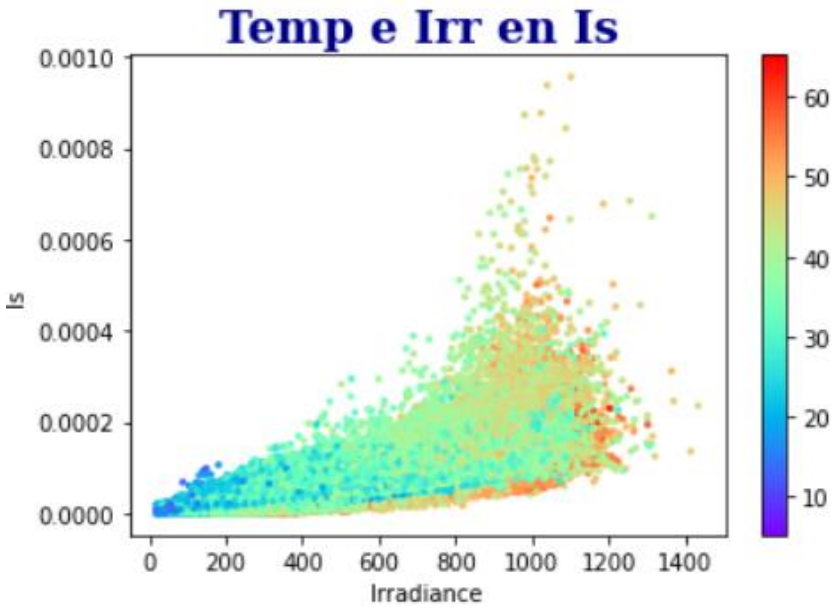

Ilustración 19: Temperatura e Irradiancia en I_s.

Se puede apreciar como la corriente de saturación inversa depende de la temperatura a la que esté sometida la placa solar, pero también de la irradiancia que reciba. Puesto que, a medida que aumenta la irradiancia, también lo hace la temperatura. Por otro lado, la relación entre la irradiancia y el parámetro Is es más complicado de relacionar utilizando únicamente el gráfico adjuntado, necesitaríamos apoyarnos para ello en determinadas ecuaciones matemáticas.

- Resistencia en serie (R_s) y resistencia en paralelo (R_{sh})

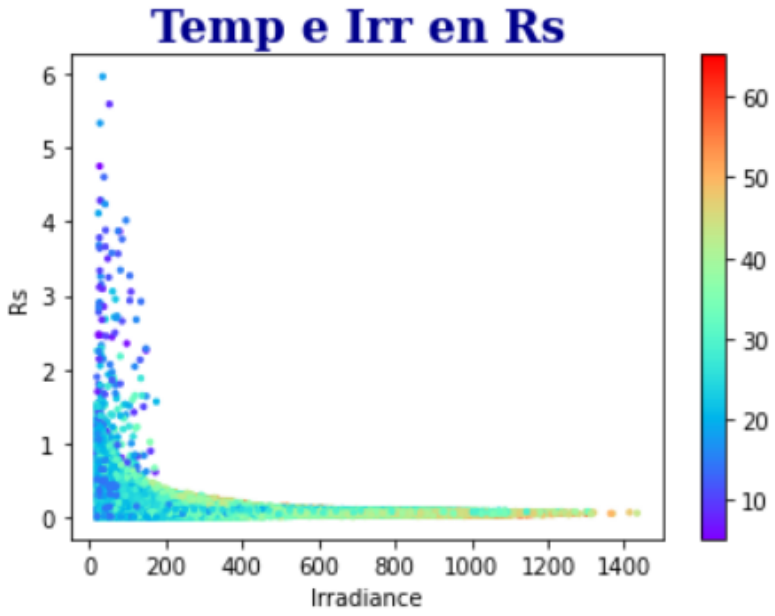

Ilustración 20: Temperatura e Irradiancia en R_s.

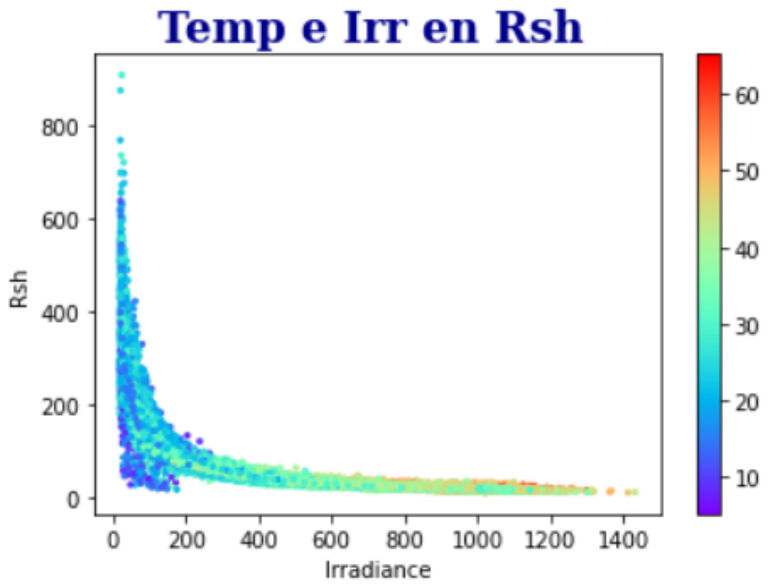

Ilustración 21: Temperatura e Irradiancia en R_{sh}.

En ambas gráficas se puede apreciar un comportamiento similar y es que a medida que se aumenta el valor de la irradiancia, menor es la resistencia que va a tener la placa solar, ya sea en serie como en paralelo. Observamos como valores muy pequeños de irradiancia van a suponer el máximo de resistencia. De la misma manera podemos relacionar la resistencia en serie y paralelo con la temperatura, puesto que son directamente proporcionales entre sí. A mayores temperaturas, menor es la resistencia de la placa solar.

- Factor de idealidad del diodo (n)

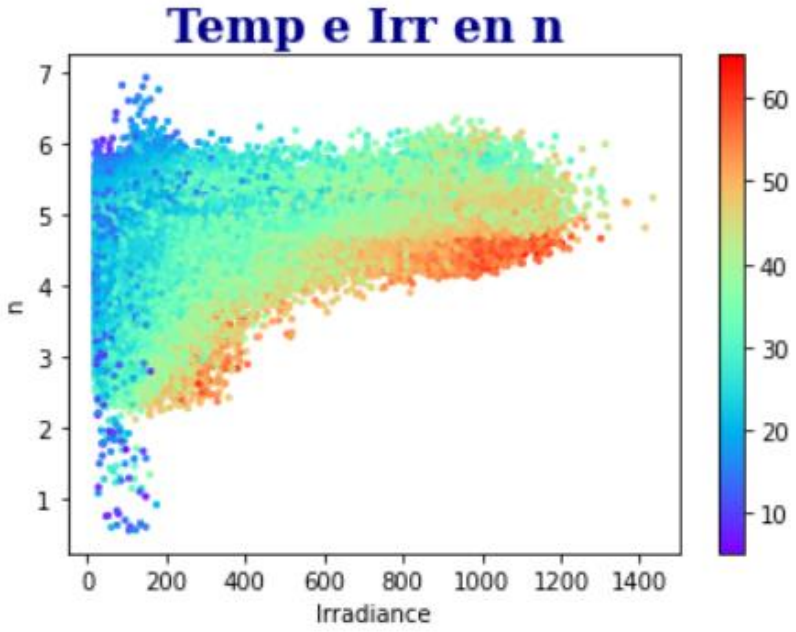

Ilustración 22: Temperatura e Irradiancia en n.

El último parámetro con el que vamos a trabajar es con el factor de idealidad del diodo, que mide lo cerca que nuestro diodo está de cumplir con la ecuación del diodo ideal.

Como podemos observar, el valor de n es inversamente proporcional al de la temperatura y es que cuanto mayor sea la temperatura recibida por la placa, menor será el factor de idealidad del diodo para una misma irradiancia.

3.3 TECNOLOGÍA ASIMICRO

La tecnología aSi se trata de módulos fotovoltaicos de silicio amorfo o también llamados de capa fina. Su uso se está haciendo cada vez más habitual, ya que tiene un menos coste por Wp (Watt pico) con respecto a los de silicio cristalino. Otra de las características a destacar es que aprovechan mejor la radiación difusa y tienen menos sensibilidad frente a la temperatura y a las sombras [20].

La primera prueba que haremos será ver las predicciones que hemos obtenido sin aplicarle ningún tipo de preprocesamiento, es decir, con los valores tal cual los hemos leído del archivo csv. Vamos a ver cómo se comporta la placa solar con la misma tecnología (*aSiMicro_03036*), en diferentes ciudades como son Cocoa y Eugene.

3.3.1 RESULTADOS SIN PREPROCESAMIENTO: COCOA

Para hacer esta primera investigación hemos cogido el modelo *aSiMicro_03036*, ubicado en la ciudad de Cocoa. Para ello, utilizamos un dataframe que está compuesto por 39037 filas.

La finalidad de este estudio es comprobar si el entrenamiento realizado ha sido o no eficiente. Para llevarlo a cabo, debemos obtener el error porcentual medio que nos proporciona información relevante, relacionando las predicciones halladas con los valores observados. Es decir, cuanto menor sea el valor del error mejores serán los resultados obtenidos con el entrenamiento de la red neuronal, pues más se acercan los valores esperados a sus valores reales. Además, conocer previamente la desviación típica de los errores hallados para cada parámetro nos ayuda a comprender el resultado del *Error Porcentual*

Absoluto Medio, pues la desviación típica cuantifica la dispersión de los datos a estudio.

En el caso de Cocoa, los datos recogidos nos proporcionan las siguientes desviaciones típicas, para cada uno de los parámetros a analizar:

$\mathbf{I_s}$	1.379598_e+30
$\mathbf{R_{sh}}$	2.759002_e-01
$\mathbf{R_s}$	1.057579_e+03
n	2.950852_e-01
$\mathbf{I_{ph}}$	3.832854_e-01

Tabla 2: Desviación típica Cocoa

Como vemos, la desviación típica de I_s es muy elevada, concretamente tiene un valor de $1.379598e^{30}$. Esto quiere decir que tiene valores muy alejados de su media aritmética y, por tanto, dispersos entre sí.

A continuación, se muestra una tabla con los errores porcentuales medios de los datos recogidos en Cocoa para cada uno de los cinco parámetros a estudio.

$\mathbf{I_s}$	2.3196_e+30%
$\mathbf{R_{sh}}$	18.81%
$\mathbf{R_s}$	7531.722%
n	9.5394%
$\mathbf{I_{ph}}$	22.8404%

Tabla 3: Error porcentual absoluto medio Cocoa

Como vemos, el error porcentual absoluto medio de algunos parámetros es muy elevado, sobre todo en el caso de I_s y R_s. Además, entre unas variables y otras existen grandes diferencias de porcentajes, lo que impide representar los cinco errores porcentuales en un mismo gráfico como si haremos en otros apartados en los que estas diferencias sean menores.

3.3.2 RESULTADOS SIN PREPROCESAMIENTO: EUGENE

Como hemos dicho con anterioridad, para hacer más completa la investigación vamos a analizar los resultados que nos ha dado el entrenamiento de la red neuronal sobre el mismo tipo de placa solar en otra región de EE.UU. Esto nos

va a permitir saber cómo se comporta esta misma tecnología en dos puntos distintos como son Cocoa y Eugene.

En este caso, el dataframe está compuesto por 43343 filas, 4306 filas más que con la ciudad de Cocoa.

Si volvemos a entrenar la red neuronal, pero con los datos de Eugene nos damos cuenta de que los valores de las desviaciones típicas son bastantes similares a las obtenidas con anterioridad:

I_s	5.823817_e+28
R_{sh}	2.633145_e-01
R_s	5.006594_e+03
n	2.592705_e-01
I_{ph}	4.085023_e-01

Tabla 4: Desviación típica Eugene

De forma análoga, calculamos los valores de los errores porcentuales absolutos medios para la ciudad de Eugene, obteniendo:

I_s	7.2247_e+28%
R_{sh}	18.1566%
R_s	1187.728%
n	11.2343%
I_{ph}	21.0034%

Tabla 5: Error porcentual absoluto medio Eugene

Como vemos, tanto las desviaciones típicas como los errores hallados en esta región son menores, aunque nada significativo.

3.3.3 RESULTADOS SIN PREPROCESAMIENTO: FUSIÓN DE DATOS DE EUGENE Y COCOA

En este apartado vamos a realizar otra prueba distinta a las anteriores. Vamos a comprobar si combinando ambos dataframe obtenemos mejores resultados que los vistos anteriormente.

Hemos usado los datos de la placa solar aSiMicro_03036 de Cocoa y de Eugene por separado. Ahora como hemos dicho, concatenamos los datos de ambas ciudades en un solo dataframe.

Por lo tanto, tendremos los 39037 datos procedentes de Cocoa y por otro lado los 43343 datos de Eugene. Es decir, 82380 filas en total. La idea de esto es hacer los mismos pasos y procedimientos que se han hecho en los apartados anteriores con cada una de las regiones por separado, pero ahora para las dos concatenadas.

Cuanto mayor sea la muestra de la población a estudio, mejores serán las predicciones que se hagan, puesto que nos aporta mayor variedad de datos reales.

Volvemos a entrenar la red neuronal con estos datos y vemos en las siguientes tablas si el error porcentual absoluto medio y la desviación típica mejoran o empeoran entrenando la red neuronal con más del doble de datos.

I_s	6.431773_e+09
R_{sh}	2.845898_e+03
R_s	1.625658_e+03
n	1.570532_e-01
I_{ph}	6.4317738_e+09

Tabla 2: Desviación típica datos concatenados

I_s	6.040133_e+09%
R_{sh}	42.7109%
R_s	6599.34299%
n	16.9991%
I_{ph}	47.8592%

Tabla 3: Error porcentual absoluto medio datos concatenados

En este ya vemos un cambio más significativo, podemos ver como en el error porcentual absoluto medio el R_s aumenta significativamente con respecto a Eugene, pero desciende y, por lo tanto, mejora con respecto a Cocoa.

Pero sin duda, donde vemos el mayor cambio es en el parámetro I_s, tanto en Cocoa como en Eugene teníamos un valor muy elevado. A pesar de que el valor sigue siendo elevado, podemos decir que ha mejorado con notoriedad.

3.3.4 FILTRADO DE VALORES ATÍPICOS (OUTLIERS)

Antes de comenzar con el filtrado, debemos conocer qué son los outliers. Pues bien, podríamos hablar de una observación anormal en una muestra estadística o serie temporal por ser distante del resto de datos, lo que puede afectar con notoriedad y de manera negativa a la estimación de los parámetros [21].

Por lo tanto, si para los valores de nuestros 5 parámetros, obtenidos mediante el método TSLLS, encontramos algunos datos alejados del resto, éstos pueden afectar negativamente al entrenamiento de la red neuronal y, por tanto, a las predicciones que se realicen. Como consecuencia, el error porcentual absoluto medio será mayor al que obtendremos si eliminamos estos valores atípicos.

Sabiendo esto, debemos comprobar si los parámetros utilizados tienen outliers o valores atípicos. Para ello, utilizaremos los gráficos de cajas y bigotes, que serán explicados de manera detallada en el apartado siguiente. Tras esto, eliminaremos aquellos valores que disten del resto de datos tomados.

Para la realización del filtrado, se han llevado a cabo dos pasos:

- Filtrado del 20% de los datos.
- Filtrado del 50% de los datos.

La idea de esto es ver que, efectivamente, cuantos más datos eliminemos de los dataframe, el error absoluto porcentual medio decrecerá. Esto no significa que la red neuronal y sus predicciones sean mejores ya que si eliminamos un porcentaje tan alto de los datos, los resultados que obtengamos estarán alejados de la realidad.

3.3.4.1 GRÁFICO DE CAJAS Y BIGOTES

Antes de comenzar con los gráficos de nuestros outliers mediante este tipo de representación, pasaremos a comentar brevemente cómo obtener este gráfico y qué información nos proporciona.

En la siguiente ilustración veremos cómo es el diagrama de cajas y bigotes, donde se muestra a qué hace referencia cada parte de este:

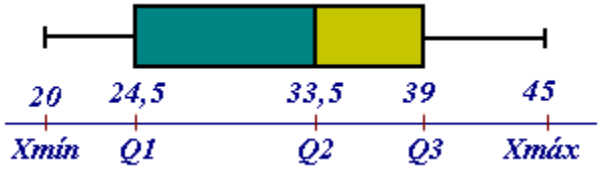

Ilustración 23: Explicación gráfico cajas y bigotes en horizontal

A las líneas que sobresalen de la caja anterior se les conoce como bigotes. Debemos tener en cuenta que estos bigotes tienen límite de prolongación, de modo que cualquier dato que no esté comprendido dentro de dichos valores serán tomados como valores atípicos. A continuación, se muestra otra figura donde se puede ver de manera más clara lo comentado anteriormente:

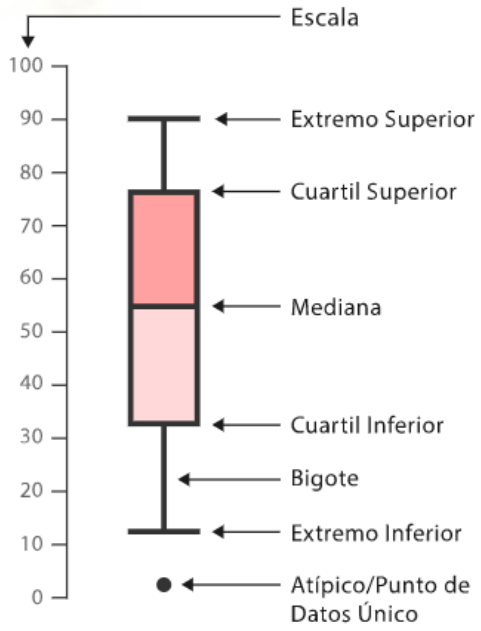

Ilustración 24: Explicación gráfico cajas y bigotes en vertical

Debemos tener en cuenta que los cuartiles son valores que dividen nuestra muestra en cuatro partes iguales. Por esta razón, se pueden utilizar para evaluar la dispersión de nuestros datos a estudio y ayudamos así a nuestra investigación.

- El primer cuartil o cuartil inferior (Q_1) nos muestra que el 25% de los datos de nuestra distribución están por debajo o son iguales a este valor.
- El segundo cuartil (Q_2) coincide con el valor central de nuestros datos y hacer referencia a la mediana de la distribución.
- El tercer cuartil o cuartil superior (Q_3) nos muestra que el 75% de los datos es inferior o igual a este valor.

Por otro lado, el gráfico de cajas y bigotes también permite observar de manera clara el rango intercuartil, que corresponde con la distancia entre el primer y tercer cuartil. Por tanto, para obtener su valor únicamente tenemos que restar $Q_3 - Q_1$, obteniendo así el 50% de los datos centrales.

Por último, vamos a ver como obtener los valores de X_{min} y X_{max}, que corresponden con el extremos inferior y superior, respectivamente.

Para calcular X_{min} restaremos 1.5 veces el valor del rango intercuartil al primer cuartil ($X_{min} = Q1 - 1.5 * \text{Rango intercuartil}$). Sin embargo, si el valor mínimo de los datos es mayor que X_{min} el extremo inferior se ubicará en el valor mínimo y X_{min} = valor mínimo.

En el caso de X_{max} se calculará sumando 1.5 veces el rango intercuartil al tercer cuartil ($X_{max} = Q_3 + 1.5 * \text{Rango intercuartil}$), pero si el valor máximo de los datos es menor que X_{max}, el extremo superior pasará a ser dicho valor. Por lo tanto, si X_{max} es mayor que el valor máximo de los datos, entonces X_{max} = valor máximo de los datos [22].

3.3.4.2 FILTRADO DE OUTLIERS DATOS: COCOA

Lo primero que se ha realizado en el trabajo para determinar si había valores atípicos ha sido comprobar los percentiles de cada parámetro, obteniendo así la siguiente tabla:

	n	Rs	Rsh	Iph	Is
count	39037.000000	3.903700e+04	39037.000000	39037.000000	3.903700e+04
mean	4.714100	1.369636e-01	77.520059	0.386943	7.094560e-05
std	0.700551	2.400420e-01	81.426583	0.296633	8.232172e-05
min	0.540672	6.353132e-07	9.547175	0.009686	1.707921e-37
2%	2.834041	2.774088e-04	15.989403	0.022437	5.606407e-08
5%	3.235009	1.079435e-03	17.555406	0.029925	2.898227e-07
10%	3.705060	5.759605e-03	19.315538	0.046352	1.412191e-06
25%	4.374533	4.196174e-02	24.372351	0.120525	9.496451e-06
50%	4.869532	5.994544e-02	47.676425	0.299980	3.652913e-05
75%	5.201397	1.227841e-01	93.890149	0.670830	1.144156e-04
80%	5.268973	1.607809e-01	114.315260	0.738902	1.345572e-04
98%	5.740310	8.812940e-01	337.961486	0.919558	2.878541e-04
max	6.934617	5.969896e+00	910.385350	1.208103	9.568511e-04

Ilustración 25: Percentiles Cocoa

En ella podemos apreciar que en algunos parámetros como en el I_s el valor mínimo es muy pequeño o que por ejemplo el valor máximo del R_{sh} es muy grande en comparación con el resto de los datos.

Esto ha llevado a la búsqueda de un método que nos proporcione información acerca de los valores atípicos que puede poseer nuestra muestra y que pueden dar lugar a equívocos en el desarrollo de la investigación. Para ello, se ha propuesto el gráfico de cajas y bigotes mencionado con anterioridad.

Sin más dilación, a continuación, se muestran los gráficos de cajas y bigotes para los datos obtenidos en la ciudad de Cocoa.

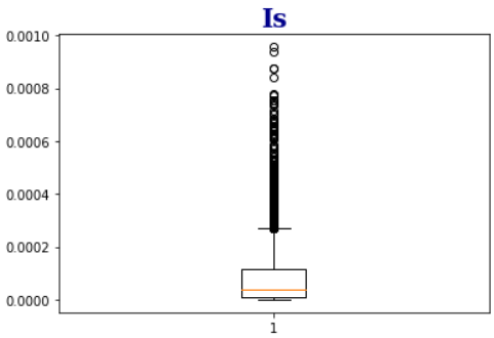

Ilustración 27: Diagrama cajas y bigotes Rsh Cocoa sin filtrar.

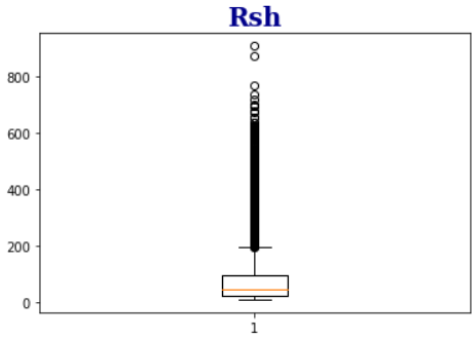

Ilustración 26: Diagrama cajas y bigotes Is Cocoa sin filtrar.

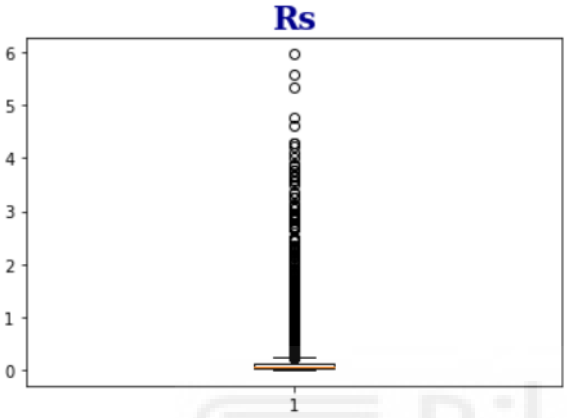

Ilustración 29: Diagrama cajas y bigotes Rs Cocoa sin filtrar

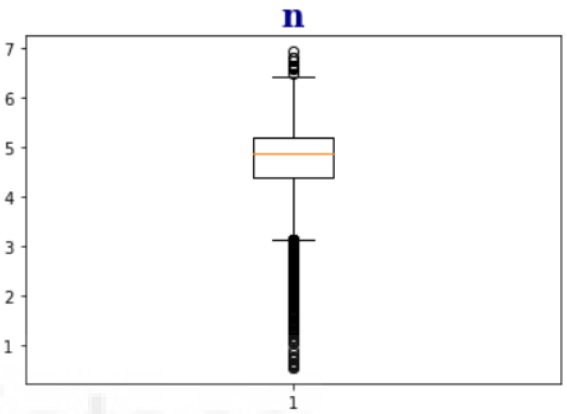

Ilustración 28: Diagrama cajas y bigotes n Cocoa sin filtrar

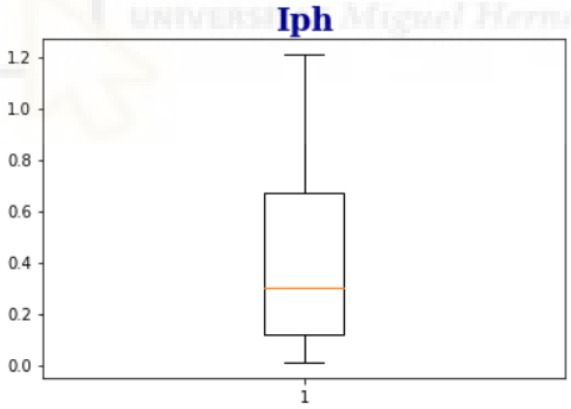

Ilustración 30: Diagrama cajas y bigotes Iph Cocoa sin filtrar

Como vemos, salvo en I_{ph} todos los parámetros tienen valores atípicos (por encima o por debajo de ambos límites) que provocarán un mayor porcentaje de error en cuanto al entrenamiento de la red neuronal. Aparentemente, el parámetro con mayor número de valores atípicos es R_s.

El siguiente paso será eliminar esos datos atípicos para mejorar nuestro entrenamiento. El problema es que si borramos un número de datos de un

parámetro también tenemos que borrar esos mismos para todos los demás parámetros. Los parámetros que se están pretendiendo eliminar son los de la salida de la red neuronal, del mismo modo es necesario eliminar esas mismas filas para los valores de entrada, ya que deben de coincidir la cantidad de datos totales de ambas.

- **FILTRADO DEL 20%:**

Con el objetivo de eliminar valores atípicos, la primera prueba que realizaremos será hacer un filtrado al 20% aproximadamente. Esto quiere decir que, si tenemos 39037 datos, eliminaremos unas 8000 filas.

Por lo que, tras realizar esta prueba nos quedamos exactamente con 31007 filas en el dataframe. A continuación, vamos a ver como quedarían los gráficos de cajas y bigotes filtrando el 20% de los datos.

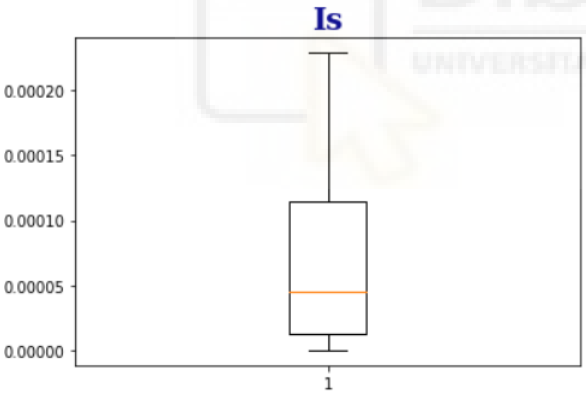

Ilustración 32: Diagrama cajas y bigotes I_s Cocoa 20% filtrado.

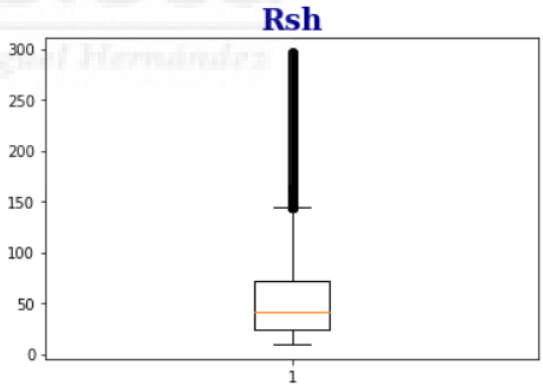

Ilustración 31: Diagrama cajas y bigotes R_{sh} Cocoa 20% filtrado.

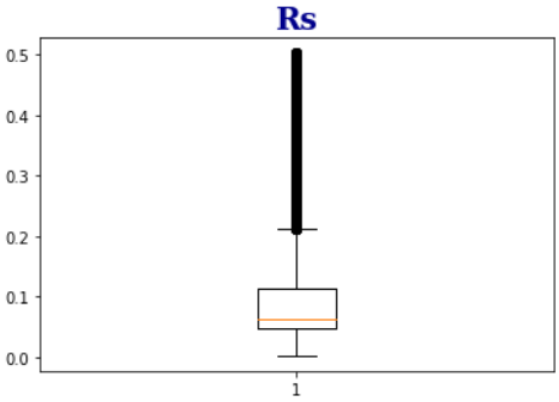

Ilustración 34: Diagrama cajas y bigotes R_s Cocoa 20% filtrado.

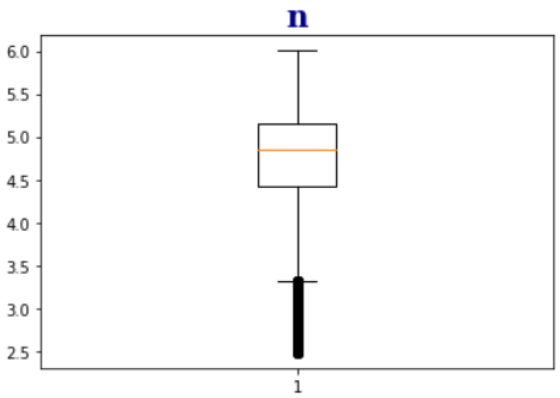

Ilustración 33: Diagrama cajas y bigotes n Cocoa 20% filtrado.

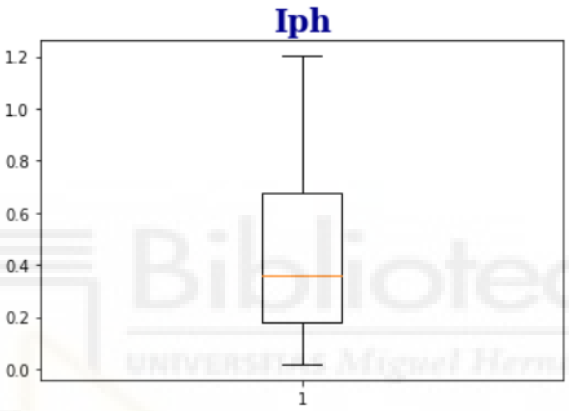

Ilustración 35: Diagrama cajas y bigotes I_{ph} Cocoa 20% filtrado.

Como podemos observar en los diagramas de cajas y bigotes, al filtrar el 20% de los datos, muchos de los outliers desaparecen. Sin embargo, en el caso de los parámetros R_s y R_{sh} seguimos teniendo bastantes. No obstante, los resultados muestran una mejoría a nivel general y, por tanto, el entrenamiento de la red neuronal podría ser más favorable.

Además, debemos tener en cuenta que para obtener las predicciones y que se ajusten a la realidad no podemos eliminar un porcentaje elevado de datos. Por ello, este primer filtrado sería adecuado a pesar de no eliminar la totalidad de datos atípicos.

	n	Rs	Rsh	Iph	Is
count	31007.000000	31007.000000	31007.000000	31007.000000	3.100700e+04
mean	4.732252	0.098438	57.662307	0.421526	6.743426e-05
std	0.589293	0.091366	49.147668	0.276641	6.375230e-05
min	2.478824	0.001940	9.547175	0.018134	9.722603e-09
2%	3.105003	0.003727	15.773748	0.042940	3.252075e-07
5%	3.551202	0.010789	17.293782	0.060101	1.242604e-06
10%	3.945467	0.029185	19.057533	0.089680	3.534092e-06
25%	4.415045	0.047526	23.743162	0.177708	1.207982e-05
50%	4.848691	0.063832	40.928699	0.357892	4.502386e-05
75%	5.150112	0.113359	71.946838	0.676975	1.146475e-04
80%	5.215204	0.138110	81.892851	0.737389	1.307652e-04
98%	5.605720	0.402493	220.758110	0.917242	2.122466e-04
max	6.006877	0.503052	296.107860	1.201627	2.290628e-04

Ilustración 36: Percentiles Cocoa 20% filtrado.

A continuación, vamos a ver como quedaría el resultado del error porcentual absoluto medio y de la desviación típica.

I_s	14.1106
R_{sh}	0.1572
R_s	7.5793
n	0.0540
I_{ph}	0.2358

Tabla 4: Desviación típica filtrado 20% Cocoa

I_s	173.4697 %
R_{sh}	25.6561 %
R_s	190.9916%
n	6.2191%
I_{ph}	18.0473 %

Tabla 5: Error porcentual absoluto medio filtrado 20% Cocoa

Como vemos, la desviación típica de todos los parámetros disminuye con la eliminación de outliers. En el caso del error porcentual absoluto medio, mejora en prácticamente todos los parámetros. En el caso de R_s e I_s, siguen siendo

elevados si los comparamos con el resto de parámetros. Aun así, el resultado ha mejorado con notoriedad.

- **FILTRADO DEL 50%:**

Repetiremos el proceso anterior, pero en este caso filtrando el 50% de los datos. Para ello, nos centraremos en eliminar outliers de los parámetros R_s e I_s que son los que mayor porcentaje de error absoluto medio tenían. De esta forma, nos quedaremos con un total de 19.701 filas.

Vamos a ver como quedarían los gráficos de cajas y bigotes tras realizar este proceso.

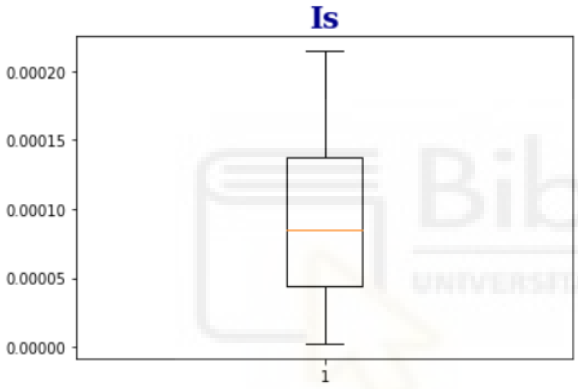

Ilustración 38: Diagrama cajas y bigotes Is Cocoa 50% filtrado.

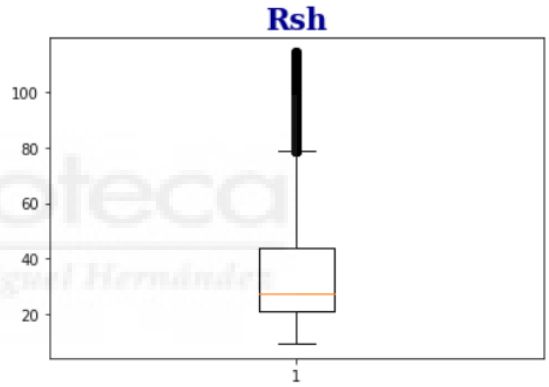

Ilustración 37: Diagrama cajas y bigotes Rsh Cocoa 50% filtrado.

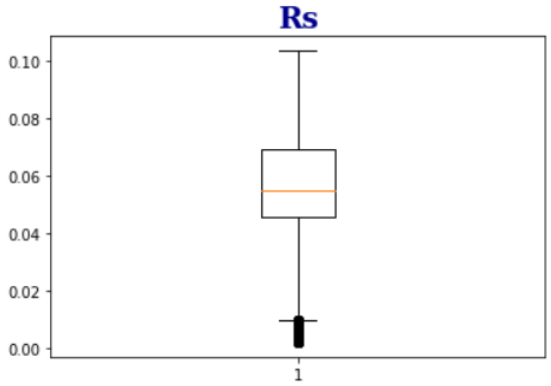

Ilustración 40: Diagrama cajas y bigotes Rs Cocoa 50% filtrado.

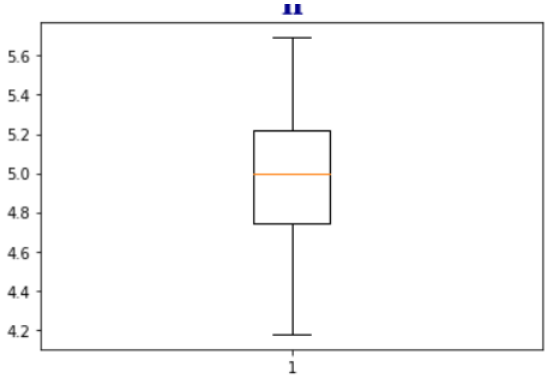

Ilustración 39: Diagrama cajas y bigotes n Cocoa 50% filtrado.

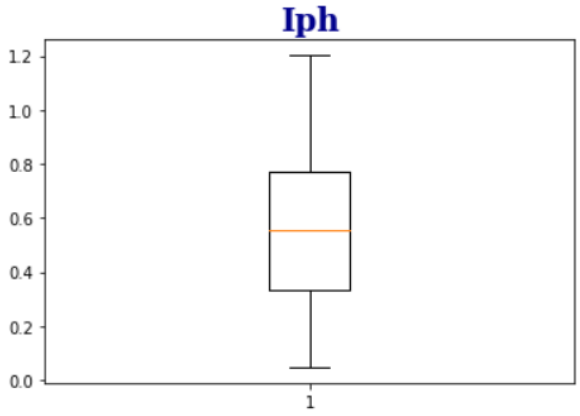

Ilustración 41: Diagrama cajas y bigotes Iph Cocoa 50% filtrado.

Observando los diagramas anteriores, podemos comprobar que el filtrado ha sido favorable para eliminar outliers tanto en R_s como en I_s.

Ahora, podemos ver cómo ha afectado este filtrado a la desviación típica y error porcentual de los cinco parámetros mediante los siguientes gráficos:

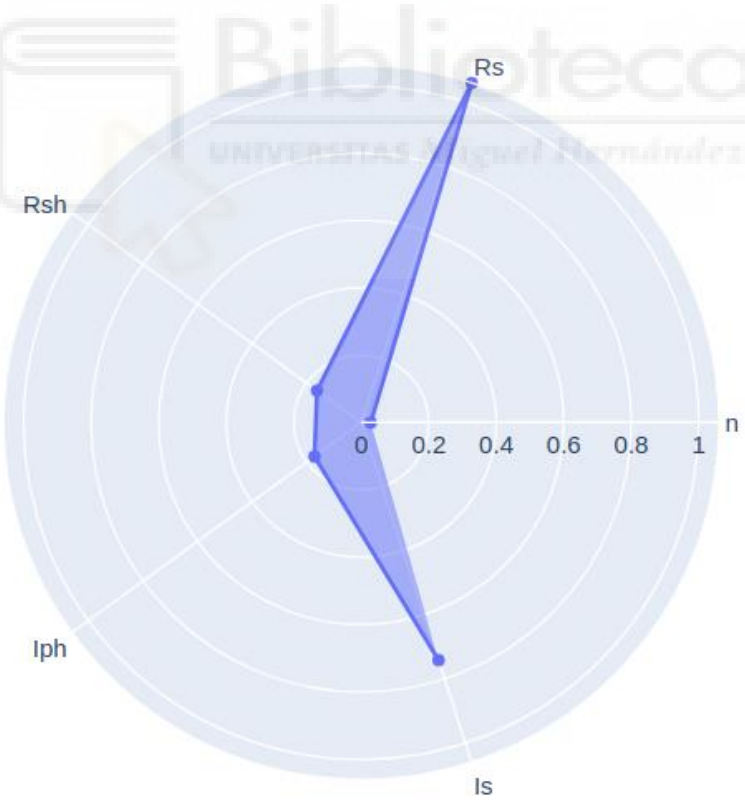

Ilustración 42: Desviación típica filtrado 50% Cocoa

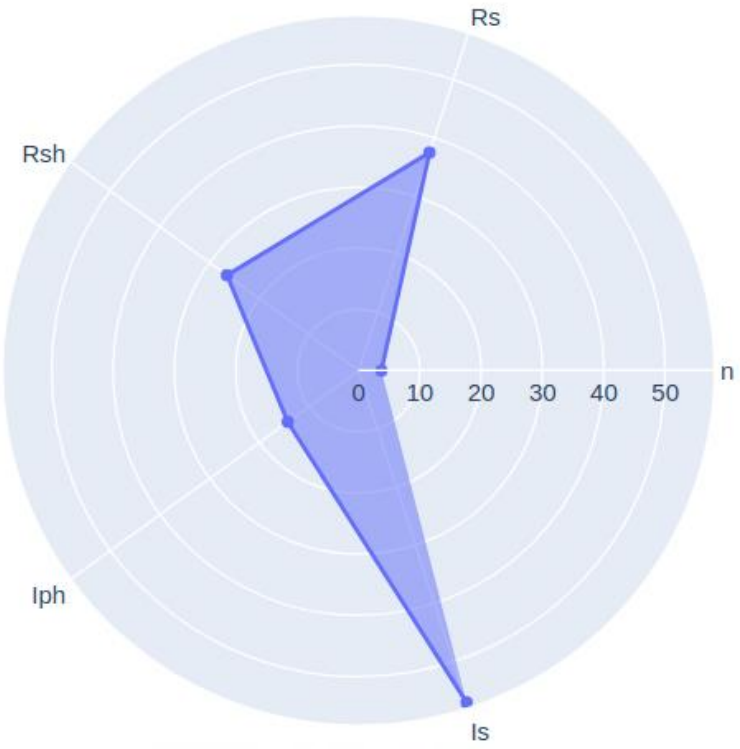

Ilustración 43: Error porcentual absoluto medio filtrado 50% Cocoa.

En este caso, sí que podemos graficar ambos resultados puesto que los valores obtenidos para los cinco parámetros son más cercanos entre sí.

En el caso de la desviación típica todos los parámetros tienen valores entre 0 y 1.1, mientras que en el filtrado anterior la desviación típica de R_s era 7.5793 y la de I_s 14.1106.

Por otro lado, también encontramos grandes modificaciones en los errores porcentuales de los cinco parámetros. Mientras que antes teníamos errores de 190.9916% y 173.4697% para los parámetros R_s e I_s, ahora todos los valores se encuentran dentro del intervalo [0,60].

Sin embargo, debemos tener en cuenta que eliminar un mayor número de datos no otorga mejores predicciones, puesto que los resultados proporcionados no serán realistas. Por tanto, no es conveniente abusar de la eliminación de las filas del dataframe a pesar de que esto mejore la desviación típica y el error porcentual absoluto medio.

De hecho, en las siguientes imágenes vamos a ver si durante los epoch, es decir, durante las iteraciones del entrenamiento la distancia entre las predicciones y los valores reales han ido convergiendo o por el contrario se han ido distanciando. Con el entrenamiento de la red neuronal lo que se pretende es minimizar la distancia entre los valores reales y las predicciones. Por lo que, vamos a ver a continuación cuáles son los valores de esa pérdida en ambos filtrados.

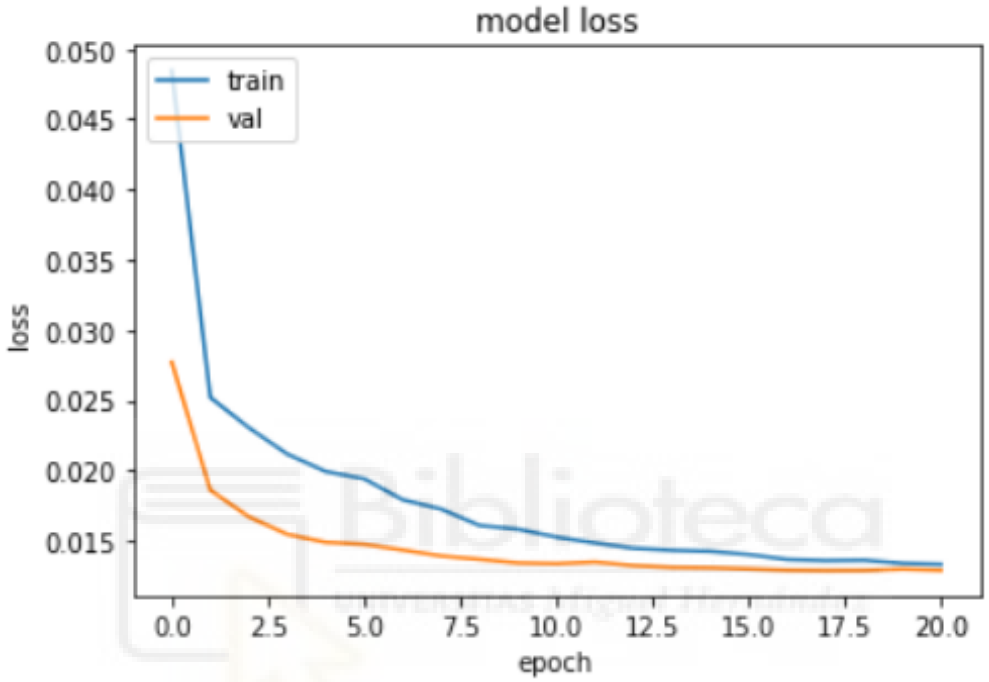

Ilustración 44: Resultado epoch Cocoa filtrado 20%

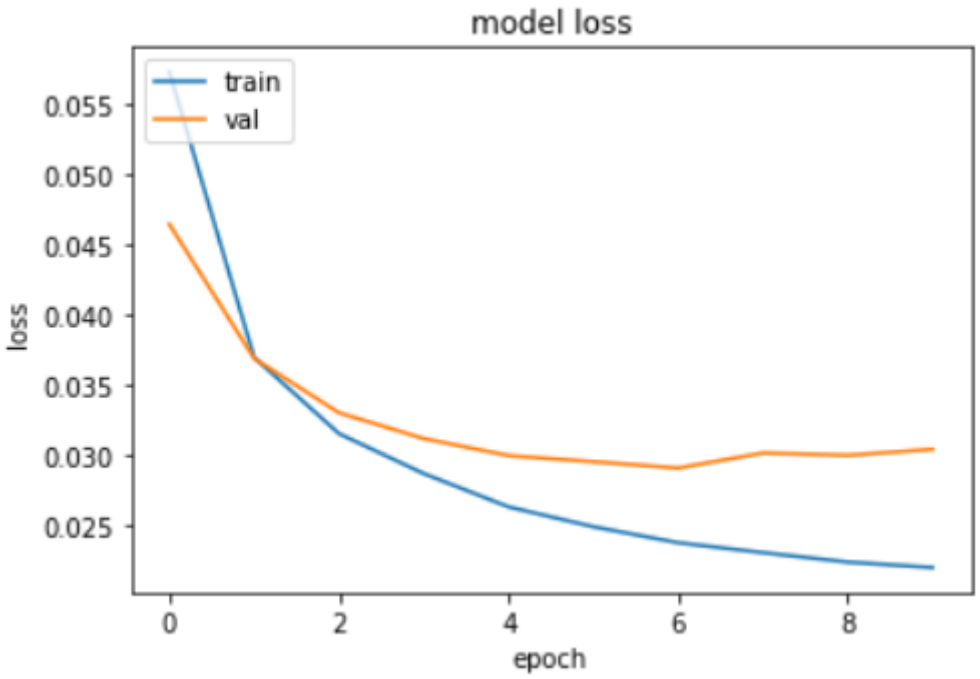

Ilustración 45: Resultado epoch Cocoa filtrado 50%

Como vemos, en el filtrado del 20% se detiene tras 21 iteraciones de entrenamiento, mientras que para el filtrado del 50% realiza 11 epoch. Esto se debe al parámetro *Early Stopping cb* o para anticipada, que se estableve a la hora de entrenar la red neuronal. Su función es detener el entrenamiento de la red neuronal cuando la pérdida de validación (val_loss) deja de disminuir. De esta manera, se evita el empeoramiento de los resultados.

Como vemos en las gráficas, el filtrado del 20% a pesar de que obtiene un error más elevado, otorga más fiabilidad en las predicciones llegando a converger. Sin embargo, el filtrado del 50% nos ha permitido profundizar en la investigación, pero no será utilizado para el entrenamiento de la red neuronal. Esto se debe a que al haber eliminado un elevado número de datos se aleja de la realidad, como se refleja en las pérdidas mostradas en la *ilustración 45.*

3.3.4.3 FILTRADO DE OUTLIERS DATOS: EUGENE

Partiendo del proceso realizado para la ciudad de Cocoa, haremos exactamente lo mismo con la finalidad de poder ampliar la investigación teniendo varios puntos distintos.

Lo primero será obtener la tabla de descripciones de los percentiles:

	n	Rs	Rsh	Iph	Is
count	43343.000000	4.334300e+04	43343.000000	43343.000000	4.334300e+04
mean	5.398153	1.667593e-01	109.775858	0.287230	8.380782e-05
std	0.762451	3.489586e-01	104.815914	0.270093	9.702266e-05
min	0.510940	1.841775e-07	4.138404	0.009733	8.991796e-47
2%	2.750271	4.228466e-04	15.095505	0.020813	1.217570e-08
5%	3.727305	1.307702e-03	16.685234	0.025322	5.080239e-07
10%	4.725228	4.194717e-03	18.498445	0.033605	6.701081e-06
25%	5.166754	4.129445e-02	28.594884	0.068438	2.066181e-05
50%	5.505689	6.549578e-02	74.957451	0.167979	5.080055e-05
75%	5.873513	1.081934e-01	153.386985	0.478405	1.181321e-04
80%	5.947607	1.369707e-01	174.913198	0.588327	1.451635e-04
98%	6.438998	1.477970e+00	425.670585	0.852557	2.928619e-04
max	7.451480	1.327236e+01	1009.238600	1.089793	1.462886e-03

Ilustración 46: Percentiles Eugene.

Al igual que en Cocoa, esta tabla nos permite obtener los percentiles que utilizaremos para las gráficas de cajas y bigotes y nos da información sobre los parámetros a estudio. No obstante, nos apoyaremos en las gráficas de cajas y bigotes para observar con más claridad los outliers.

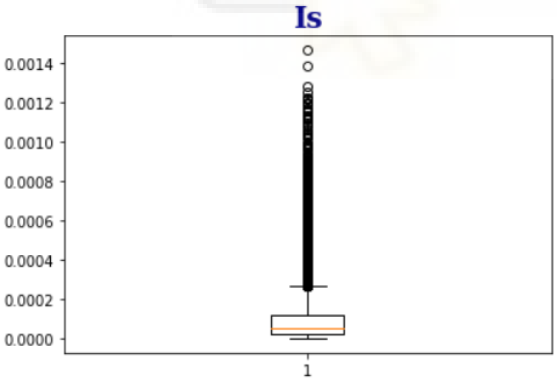

Ilustración 48: Diagrama cajas y bigotes Is Eugene sin filtrar.

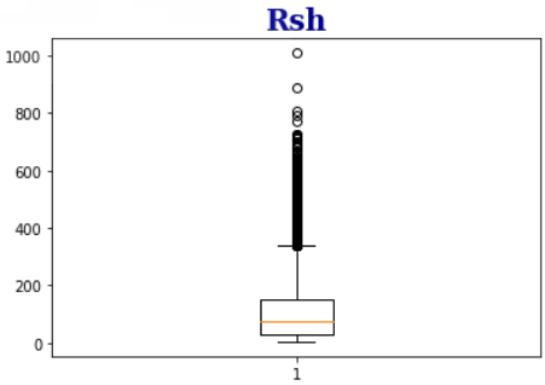

Ilustración 47: Diagrama cajas y bigotes Rsh Eugene sin filtrar.

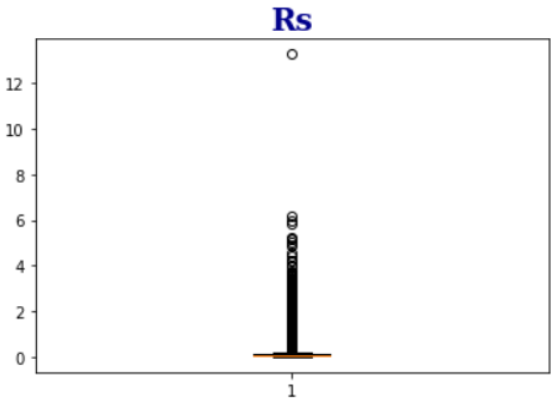

Ilustración 50: Diagrama cajas y bigotes Rs Eugene sin filtrar.

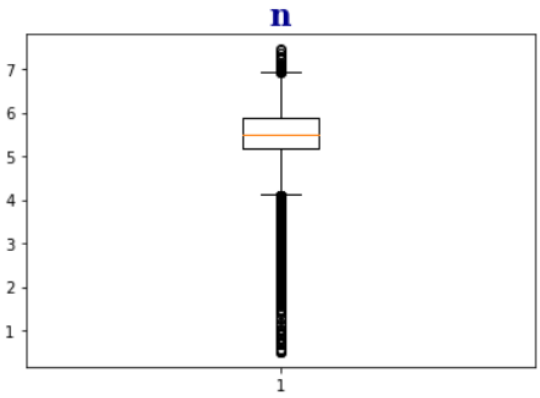

Ilustración 49: Diagrama cajas y bigotes n Eugene sin filtrar.

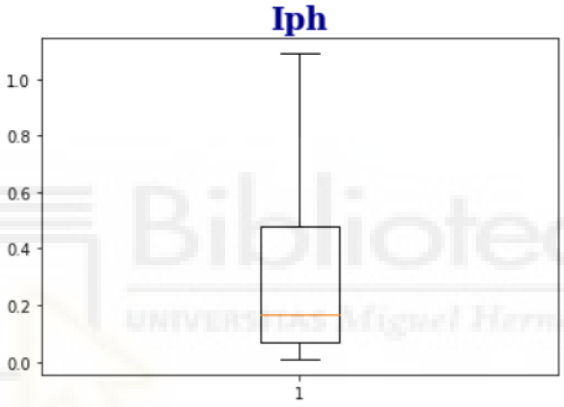

Ilustración 51: Diagrama cajas y bigotes Iph Eugene sin filtrar.

Los diagramas anteriores, muy similares a los obtenidos con Cocoa, nos muestran con claridad los valores atípicos encontrados en nuestros cinco parámetros. Además, anteriormente hemos visto que los parámetros Rs e Is son los que mayor error porcentual obtienen, hecho que se verifica con la cantidad de outliers que presentan.

- **FILTRADO DEL 20%:**

En este caso tenemos 43343 filas, por lo que filtraremos unas 8668. Tras ello, obtendremos la siguiente tabla de percentiles:

	n	Rs	Rsh	Iph	Is
count	35738.000000	35738.000000	35738.000000	35738.000000	3.573800e+04
mean	5.457420	0.114431	87.789803	0.312742	8.267947e-05
std	0.532534	0.149398	76.094381	0.262276	7.036526e-05
min	2.842997	0.001731	8.437375	0.018484	7.080077e-09
2%	3.962377	0.002784	14.986316	0.031715	1.405212e-06
5%	4.574646	0.005349	16.543251	0.039217	5.244408e-06
10%	4.890757	0.016150	18.289869	0.051317	1.121143e-05
25%	5.180048	0.054970	26.260424	0.093058	2.595371e-05
50%	5.476884	0.068541	62.534688	0.208031	5.970373e-05
75%	5.832310	0.105844	124.016548	0.518963	1.246464e-04
80%	5.906406	0.126231	141.631738	0.609623	1.482025e-04
98%	6.350646	0.680529	306.860583	0.844816	2.469975e-04
max	6.498275	1.090689	354.107920	1.089793	2.716227e-04

Ilustración 52: Percentiles Eugene 20%.

A continuación, vamos a ver las gráficas de cajas y bigotes obtenidas tras quedarnos con 35738 filas de datos.

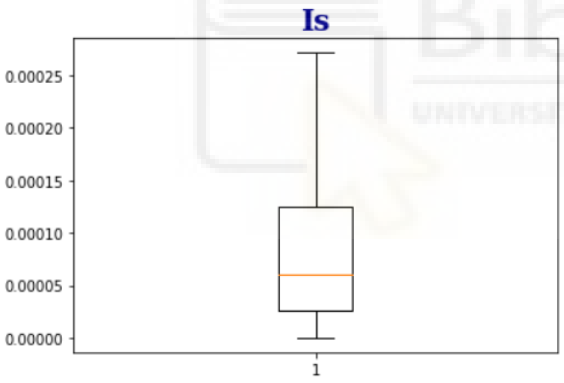

Ilustración 54: Diagrama cajas y bigotes Is Eugene 20% filtrado.

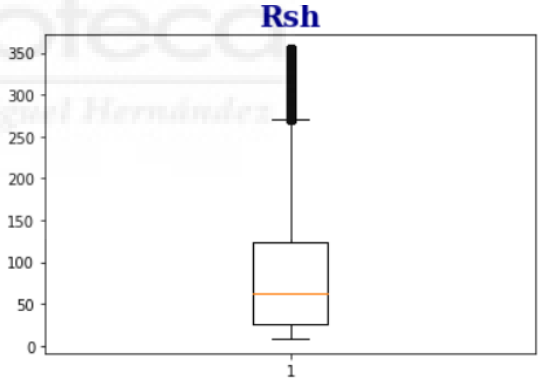

Ilustración 53: Diagrama cajas y bigotes Rsh Eugene 20% filtrado.

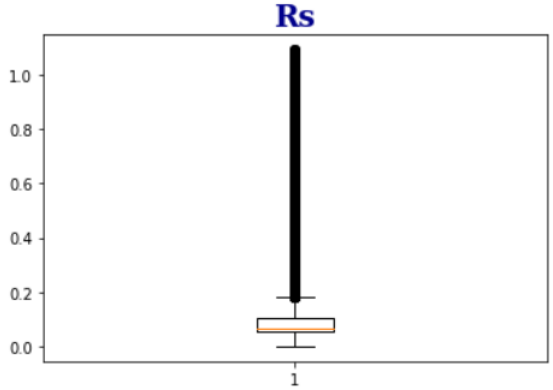

Ilustración 56: Diagrama cajas y bigotes Rs Eugene 20% filtrado.

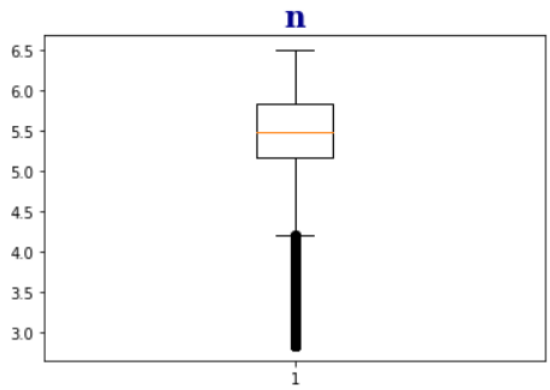

Ilustración 55: Diagrama cajas y bigotes n Eugene 20% filtrado.

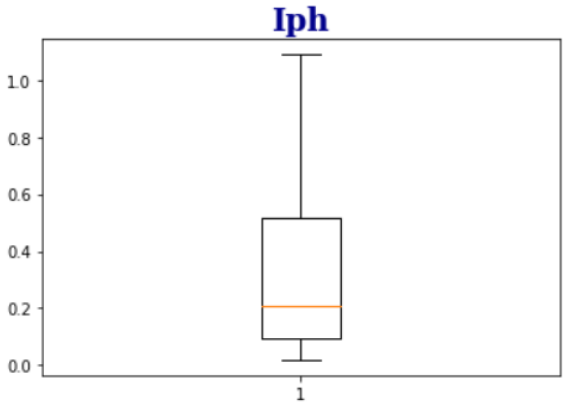

Ilustración 57: Diagrama cajas y bigotes Iph Eugene 20% filtrado.

Como vemos en los diagramas de cajas y bigotes, al hacer un filtrado del 20% se eliminan muchos de los valores atípicos. Por ejemplo, en el caso de Is se han eliminado todos aquellos valores que afectan negativamente al error porcentual y, para el resto de parámetros, se han disminuido los outliers.

Por tanto, tras este reajuste de datos, los valores para la desviación típica y el error porcentual absoluto medio serán los siguientes:

I_s	53.1537
R_{sh}	0.1190
R_s	10.6842
n	0.0792
I_{ph}	0.4014

Tabla 2: Desviación típica filtrado Eugene

I_s	396.5421%
R_{sh}	14.5317%
R_s	280.4551%
n	7.3293%
I_{ph}	24.0971%

Tabla 3: Error porcentual absoluto medio filtrado Eugene

Como en el caso de Cocoa, se ha obtenido una mejoría y por lo tanto, el error porcentual absoluto medio es inferior.

- **FILTRADO DEL 50%:**

Nuevamente, aumentaremos el filtrado de los datos al 50%. Como ya sabemos, esto nos permitirá apreciar con mayor claridad las gráficas de cajas y bigotes, al eliminar muchos de los valores atípicos.

Tras este filtrado nos quedamos con 21216 filas. Vamos a ver como quedarían los gráficos de cajas y bigotes.

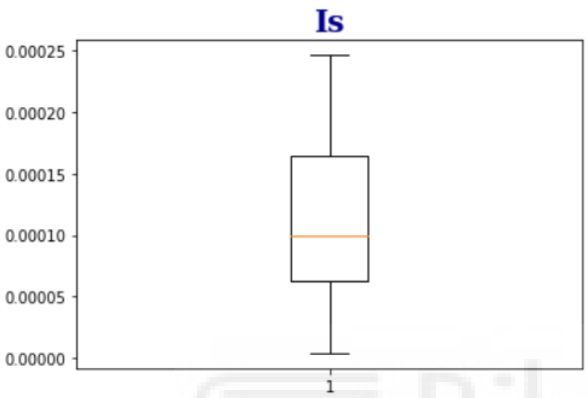

Ilustración 59: Diagrama cajas y bigotes Is Eugene 50% filtrado.

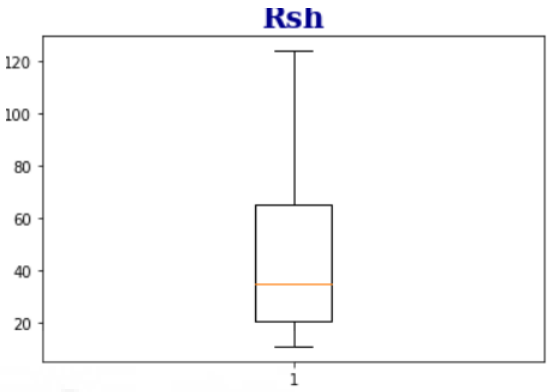

Ilustración 58: Diagrama cajas y bigotes Rsh Eugene 50% filtrado.

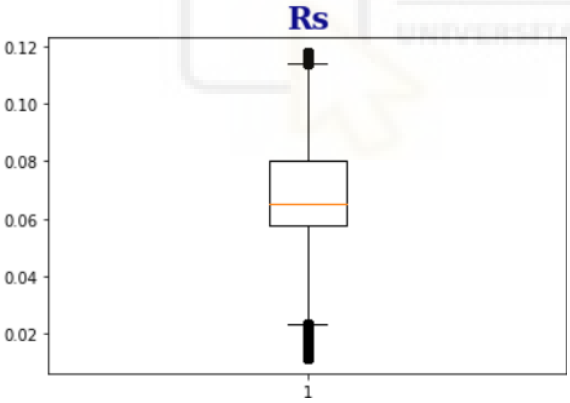

Ilustración 61: Diagrama cajas y bigotes Rs Eugene 50% filtrado.

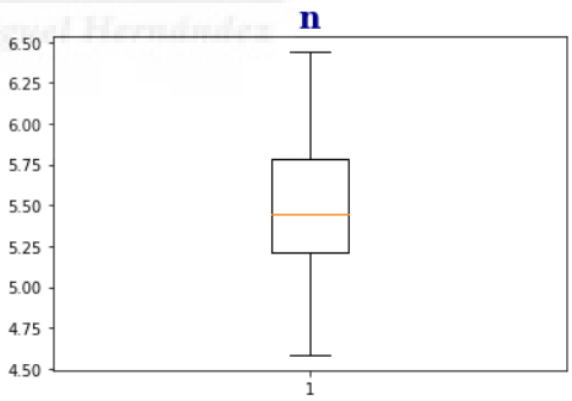

Ilustración 60: Diagrama cajas y bigotes n Eugene 50% filtrado.

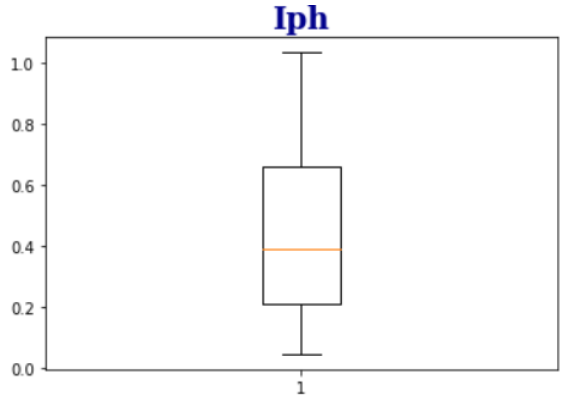

Ilustración 62: Diagrama cajas y bigotes Iph Eugene 50% filtrado.

Gracias a este filtrado, se consiguen eliminar casi todos los outliers, reduciendo el error porcentual absoluto medio y desviación típica como se muestra a continuación:

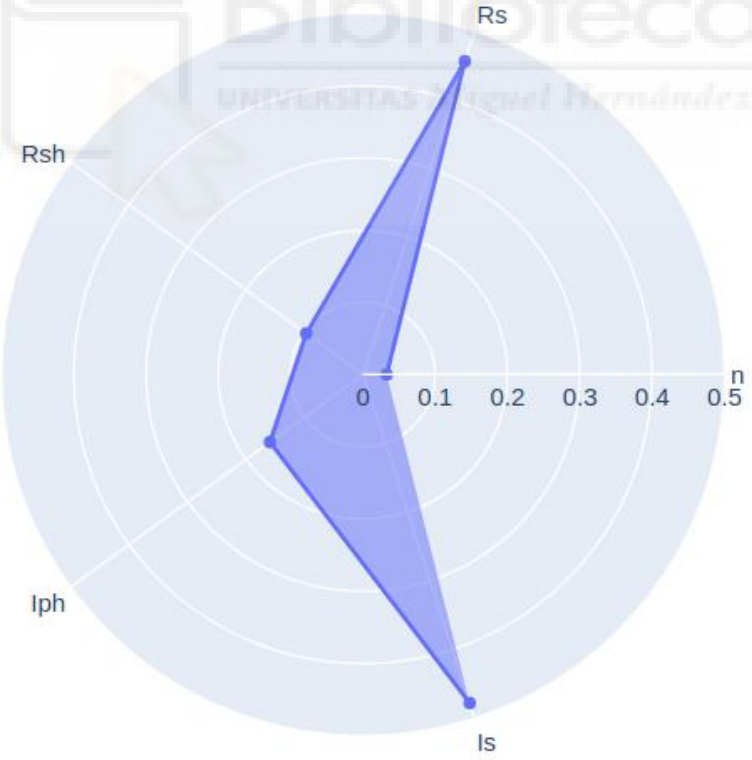

Ilustración 63: Desviación típica filtrado 50% Eugene.

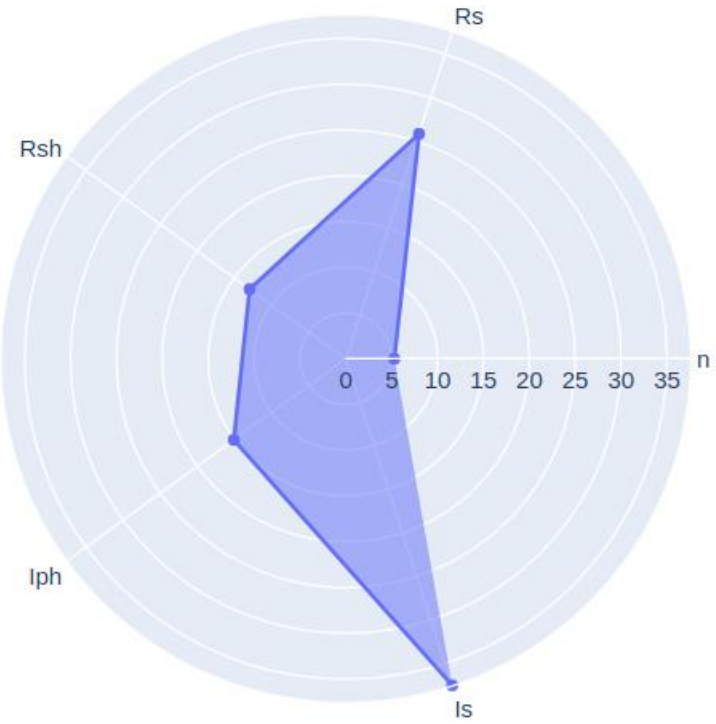

Ilustración 64: Error porcentual absoluto medio Eugene filtrado 50%

Como vemos, en este caso la desviación típica de todos los parámetros pertenece al intervalo [0,0.5] mientras que los errores están situados dentro del intervalo [0,40], mejorando considerablemente los resultados obtenidos con un filtrado del 20%.

Si nuevamente analizamos cómo varía la pérdida con el número de iteraciones del entrenamiento de la red neuronal, comprobamos que ocurre lo siguiente:

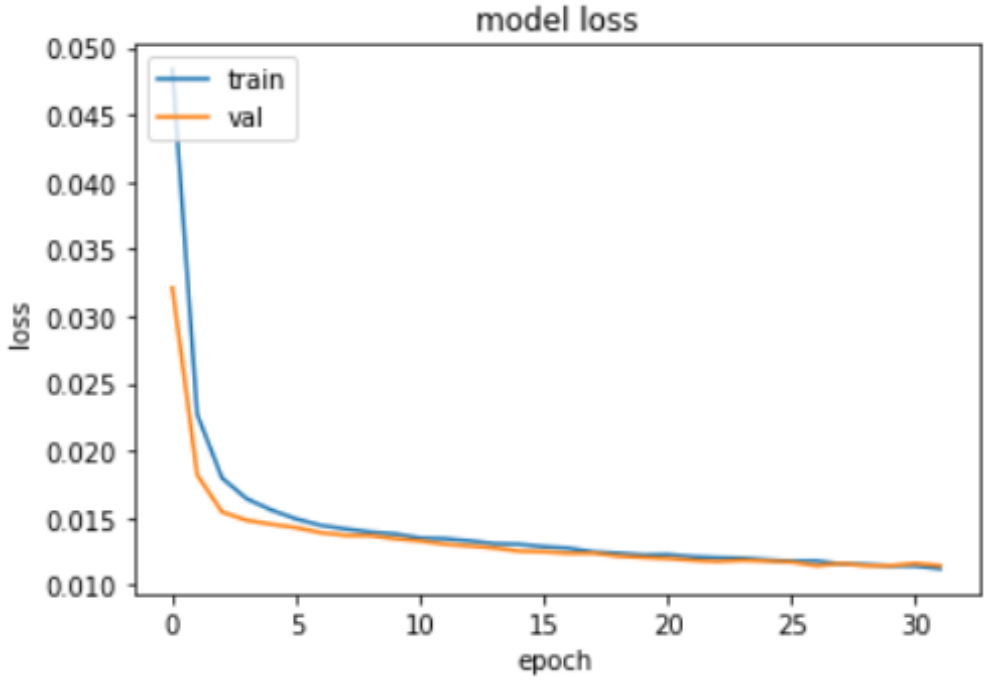

Ilustración 65: Resultado epoch Eugene filtrado 20%

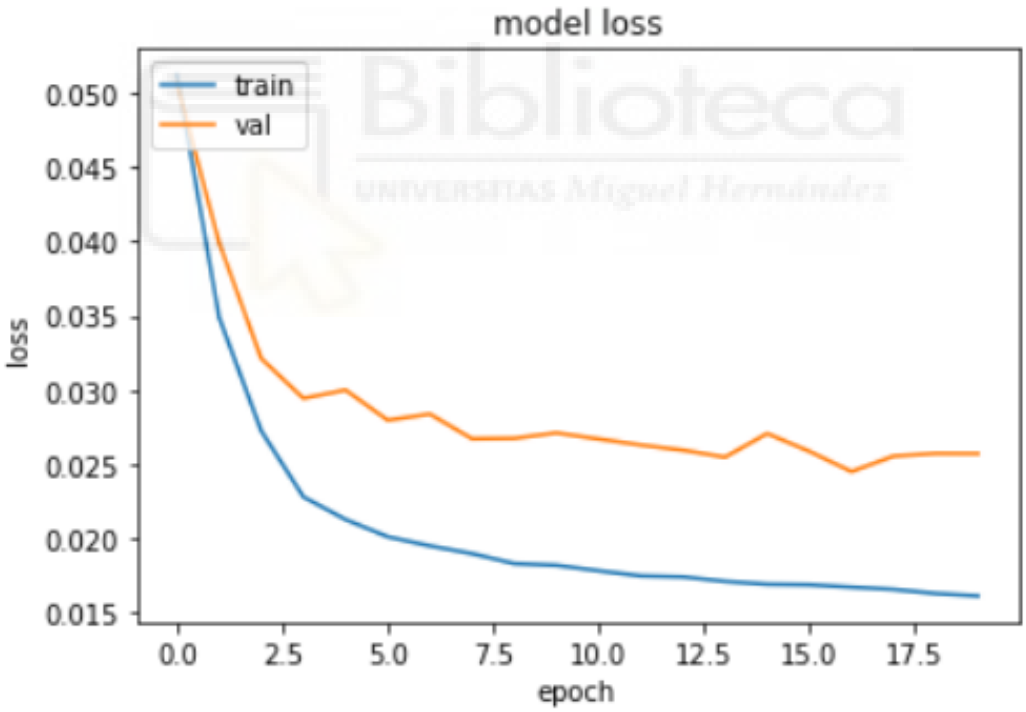

Ilustración 66: Resultado epoch Cocoa filtrado 50%

Nuevamente se comprueba que el filtrado del 20% de los datos proporciona resultados más eficientes, pues el entrenamiento de la red logra minimizar la distancia entre los valores reales y las predicciones obtenidas. Sin embargo, en el filtrado del 50% a medida que se aumenta el número de iteraciones la pérdida es mayor.

3.3.4.4 FILTRADO DE OUTLIERS DATOS: FUSIÓN DE DATOS DE EUGENE Y COCOA.

Ahora, realizaremos el filtrado de outliers, concatenando los datos de Cocoa y Eugene. Para ello, obtendremos la tabla de percentiles que, posteriormente, utilizaremos para la realización de los diagramas de cajas y bigotes, al igual que hemos hecho con las regiones de Cocoa y Eugene por separado:

	n	Rs	Rsh	Iph	Is
count	82380.000000	8.238000e+04	82380.000000	82380.000000	8.238000e+04
mean	5.074004	1.526402e-01	94.490963	0.334481	7.771286e-05
std	0.809367	3.026434e-01	95.820011	0.287325	9.058248e-05
min	0.510940	1.841775e-07	4.138404	0.009686	8.991796e-47
2%	2.791696	3.429091e-04	15.452887	0.021330	2.680155e-08
5%	3.364027	1.194097e-03	17.095007	0.027025	3.390737e-07
10%	4.005662	4.741931e-03	18.866692	0.037757	2.756759e-06
25%	4.742402	4.174983e-02	25.796404	0.084942	1.473720e-05
50%	5.202006	6.362299e-02	59.287226	0.226100	4.504381e-05
75%	5.600084	1.144029e-01	128.913860	0.583768	1.161439e-04
80%	5.715446	1.482128e-01	151.487168	0.672743	1.392355e-04
98%	6.336026	1.269994e+00	394.139305	0.896639	2.900476e-04
max	7.451480	1.327236e+01	1009.238600	1.208103	1.462886e-03

Ilustración 67: Percentiles Eugene y Cocoa.

Como era de esperar, al concatenar ambos dataframe tenemos los valores máximos y mínimos de ambos, dejando así más valores atípicos para filtrar.

El procedimiento será el mismo, por lo que no será necesario ser volver a desarrollar con detalle la explicación. No obstante, adjuntamos a continuación las gráficas antes de filtrar los datos para posteriormente hacer la comparativa tras el filtrado de outliers.

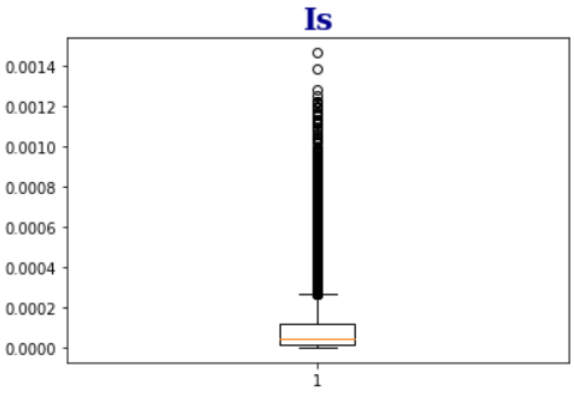

Ilustración 69: Diagrama cajas y bigotes Is Eugene y Cocoa sin filtrar.

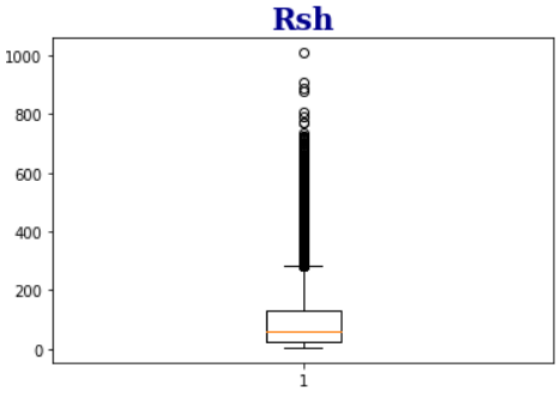

Ilustración 68: Diagrama cajas y bigotes Rsh Eugene y Cocoa sin filtrar.

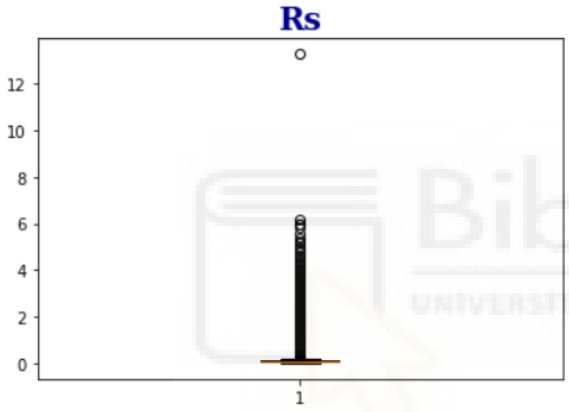

Ilustración 71: Diagrama cajas y bigotes Rs Eugene y Cocoa sin filtrar.

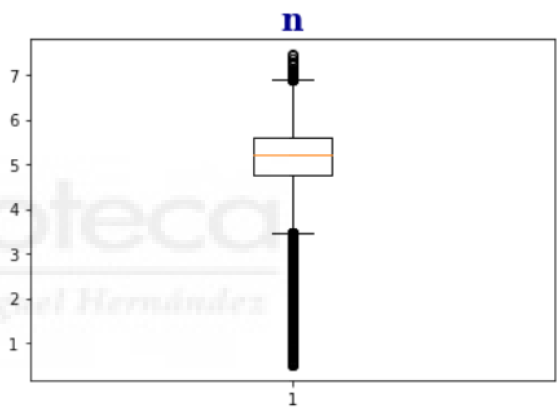

Ilustración 70: Diagrama cajas y bigotes n Eugene y Cocoa sin filtrar.

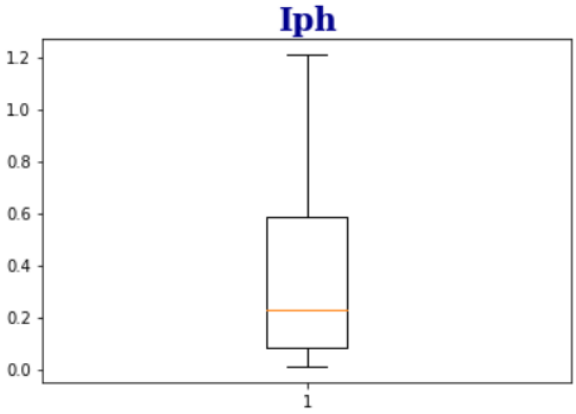

Ilustración 72: Diagrama cajas y bigotes Iph Eugene y Cocoa sin filtrar.

Como podemos ver, aparentemente los gráficos son similares a los que teníamos con ambas ciudades por separado. Siguiendo el mismo procedimiento que antes

realizaremos el filtrado de outliers eliminando datos por encima o por debajo de nuestro diagrama, dependiendo de donde estén situados los valores atípicos.

- **FILTRADO DEL 20%:**

La suma de los datos es, por un lado las 43343 filas de Eugene y por otro las 39037 filas de Cocoa. Por lo tanto, tendremos un total de 82380 filas.

Esto quiere decir que si queremos filtrar el 20% de nuestros datos tendremos que eliminar aproximadamente 16500 filas. Concretamente en esta prueba nos hemos quedado con 65043 filas de datos.

A continuación, veremos las gráficas una vez filtrado este 20%.

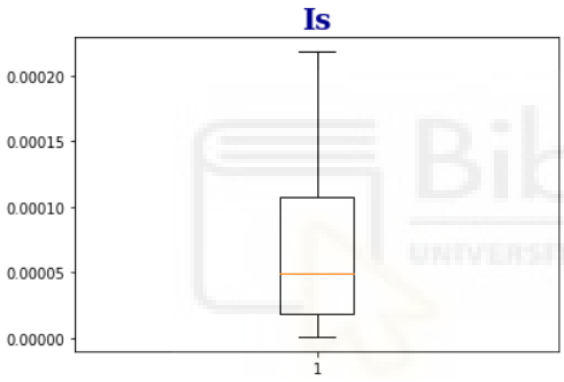

Ilustración 74: Diagrama cajas y bigotes Is Eugene y Cocoa 20% filtrado.

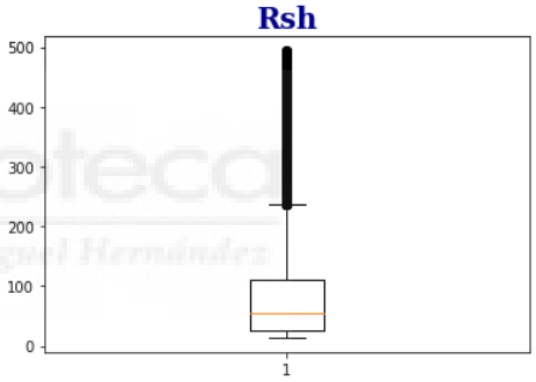

Ilustración 73: Diagrama cajas y bigotes Rsh Eugene y Cocoa 20% filtrado.

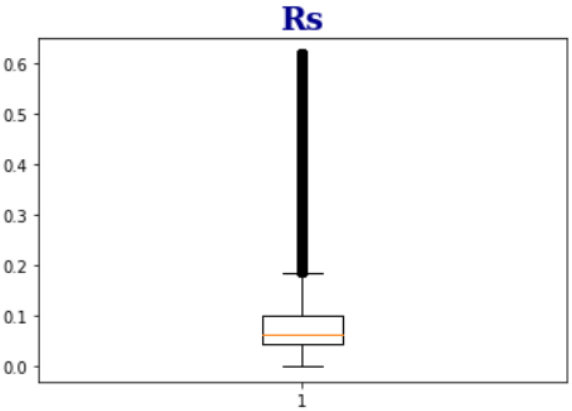

Ilustración 76: Diagrama cajas y bigotes Rs Eugene y Cocoa 20% filtrado.

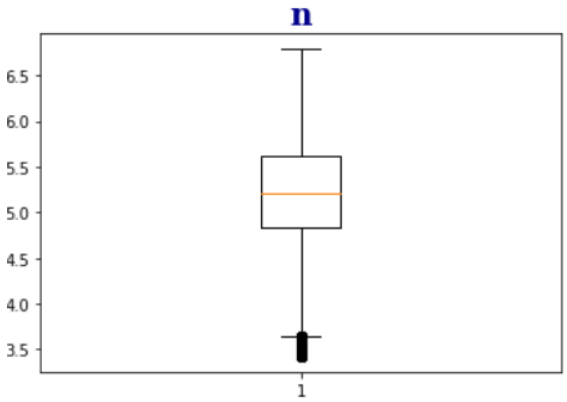

Ilustración 75: Diagrama cajas y bigotes n Eugene y Cocoa 20% filtrado.

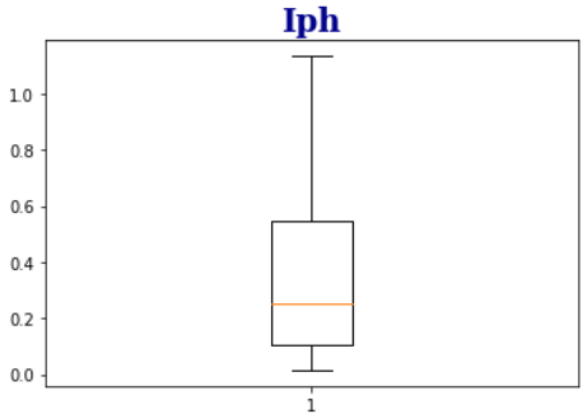

Ilustración 77: Diagrama cajas y bigotes Iph Eugene y Cocoa 20% filtrado.

Como venimos viendo en los dos apartados anteriores, los diagramas de cajas y bigotes mejoran tras hacer el filtrado sobre todo en los parámetros I_s y n.

Por lo que, como ya se adelantaba en los diagramas anteriores, los percentiles de los cinco parámetros tras el filtrado son los siguientes:

	n	Rs	Rsh	Iph	Is
count	65043.000000	65043.000000	65043.000000	65043.000000	6.504300e+04
mean	5.192144	0.091776	87.176916	0.336439	6.860566e-05
std	0.601658	0.092978	88.029737	0.267006	5.944686e-05
min	3.423330	0.001135	13.684176	0.015014	9.366126e-07
2%	3.841314	0.001875	15.622238	0.024909	2.187654e-06
5%	4.119989	0.003562	17.289656	0.034392	4.456608e-06
10%	4.363311	0.010280	19.329399	0.051373	8.038962e-06
25%	4.827656	0.045515	26.657428	0.106040	1.885165e-05
50%	5.209923	0.064966	54.738911	0.250686	4.937291e-05
75%	5.612258	0.102097	110.356755	0.548110	1.073511e-04
80%	5.729006	0.119541	131.834116	0.627019	1.247066e-04
98%	6.321765	0.417040	376.422472	0.880667	2.064280e-04
max	6.788094	0.619276	493.382170	1.134737	2.181046e-04

Ilustración 78: Percentiles Eugene y Cocoa filtrado 20%.

Veamos a continuación el error porcentual absoluto medio y la desviación típica obtenida tras el filtrado del 20%.

I_s	5.5558
R_{sh}	0.1447
R_s	9.4645
n	0.0840
I_{ph}	0.9409

Tabla 8: Desviación típica filtrado Cocoa y Eugene

I_s	355.2492 %
R_{sh}	29.7361 %
R_s	256.4496 %
n	13.2979%
I_{ph}	38.9519 %

Tabla 9: Error porcentual absoluto medio filtrado Cocoa y Eugene

Si comparamos este resultado con el filtrado por separado de ambas ciudades, también eliminando el 20% de los datos, vemos que sufren modificaciones relevantes. Si lo comparamos con la ciudad de Cocoa, aunque las desviaciones típicas mejoran en todos sus parámetros, los errores encontrados son mucho más elevados. Por otro lado, para la ciudad de Eugene decrece considerablemente la desviación típica de los parámetros I_s y R_s que pasan de 53.1537 a 5.5558 y de 10.6842 a 9.4645, respectivamente. En cuanto a los errores, aumentan un poco en todos los parámetros a excepción de Is, que pasa de tener un error porcentual del 396.5421% al 355.2492%.

Sin embargo, al comparar estos resultados con los obtenidos antes de realizar el filtrado de datos, el cambio es significativo. Las desviaciones típicas de los cinco parámetros han disminuido rotundamente, lo que se ha reflejado además en los valores de sus errores. Absolutamente todos los parámetros han sufrido modificaciones favorables, pero entre ellos destacaremos la reducción producida en el parámetro I_s que se ha pasado de un error porcentual de $6.040133e^{9}$% a tener un error del 355.2492%.

- **FILTRADO DEL 50%:**

Continuemos ahora filtrando el 50% de los datos, eliminando aquellos valores que numéricamente son distantes del resto y que pueden afectar de manera negativa a nuestra muestra. De esta manera, pasaremos de un total de 82380 filas a trabajar con tan solo 41200.

Vemos a continuación los gráficos de cajas y bigotes obtenidos tras el filtrado:

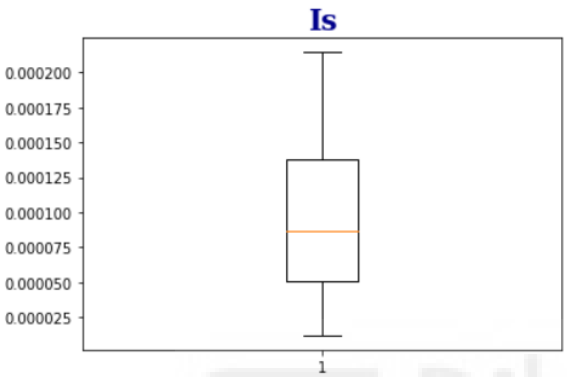

Ilustración 80: Diagrama cajas y bigotes Is Eugene y Cocoa 50% filtrado.

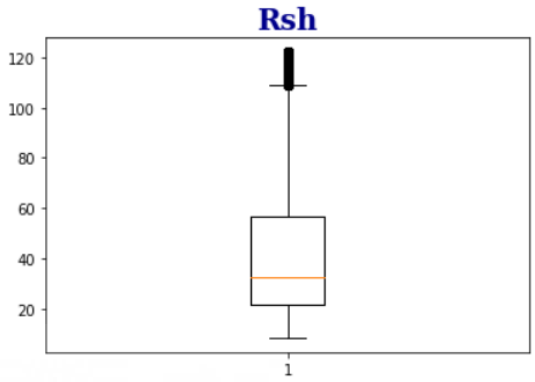

Ilustración 79: Diagrama cajas y bigotes Rsh Eugene y Cocoa 50% filtrado.

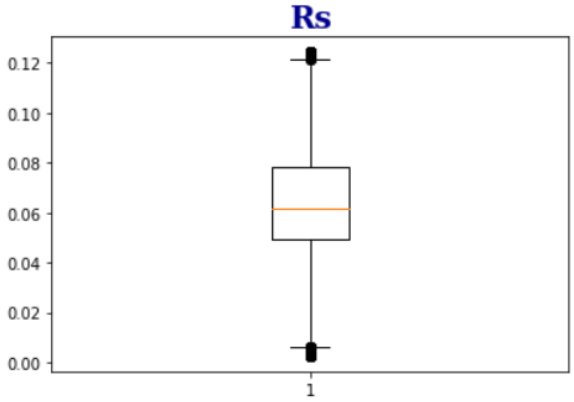

Ilustración 82: Diagrama cajas y bigotes Rs Eugene y Cocoa 50% filtrado.

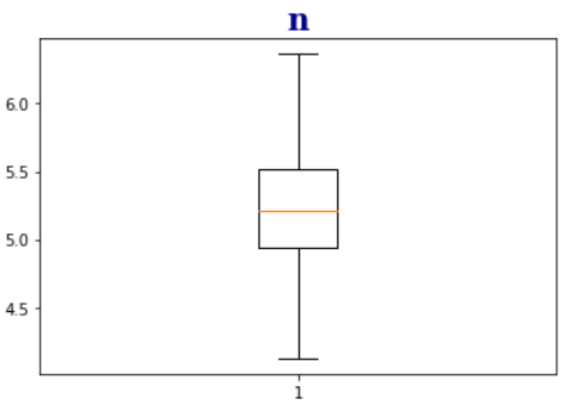

Ilustración 81: Diagrama cajas y bigotes n Eugene y Cocoa 50% filtrado.

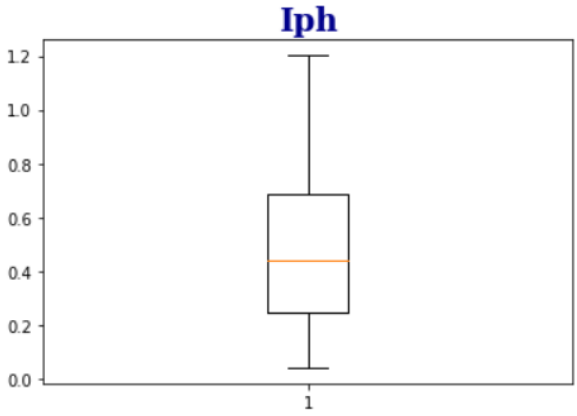

Ilustración 83: Diagrama cajas y bigotes Iph Eugene y Cocoa 50% filtrado.

Una vez más, tras eliminar la mitad de los datos, ya sea por los percentiles superiores o por los inferiores, vemos que los diagramas se ven de manera casi idílica eliminando prácticamente todos los outliers de la muestra.

Esto hace que los resultados obtenidos sufran una mejoría, como vemos a continuación:

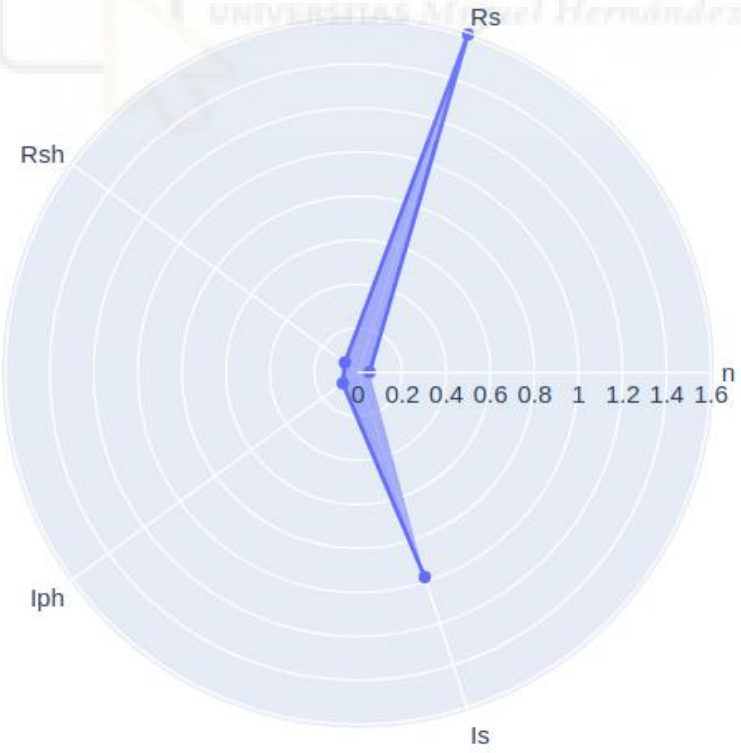

Ilustración 84: Desviación típica filtrado 50% Eugene y Cocoa.

Como vemos en el caso de la desviación típica todos los valores se encuentran dentro del intervalo [0,1.6], con un parámetro de valor prácticamente nulo.

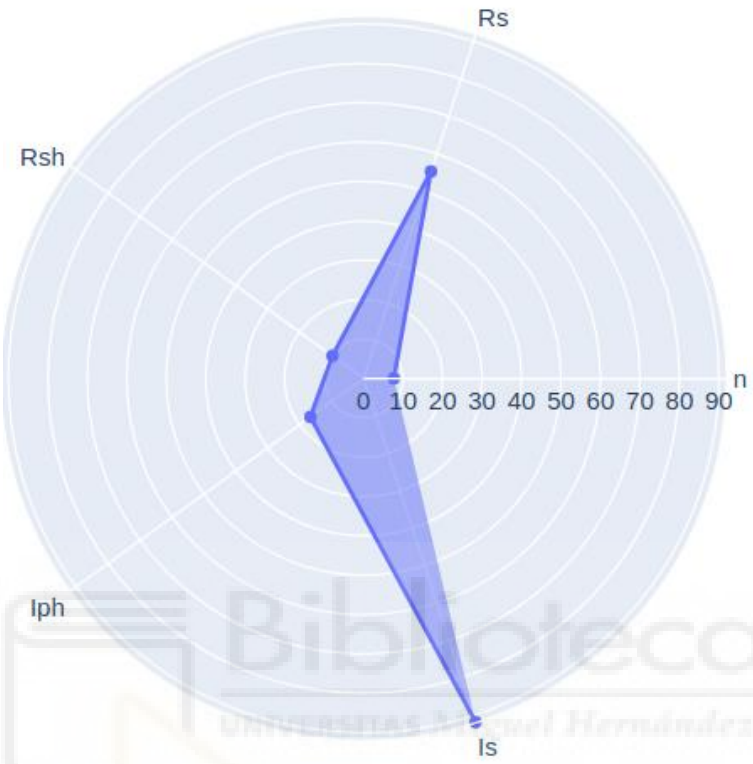

Ilustración 85: Error porcentual absoluto medio filtrado 50% Eugene y Cocoa.

En cuanto a los errores, todos se encuentran dentro del intervalo [0%,90%], lo que supone una reducción considerable. De hecho, los parámetros Is y Rs tenían errores del 355.2492% y 256.4496% respectivamente con un filtrado del 20%.

Nos apoyaremos a continuación en el modelo de pérdidas para ver los beneficios que nos proporciona el entrenamiento de la red neuronal si filtramos el 20% de los datos y si filtramos el 50%.

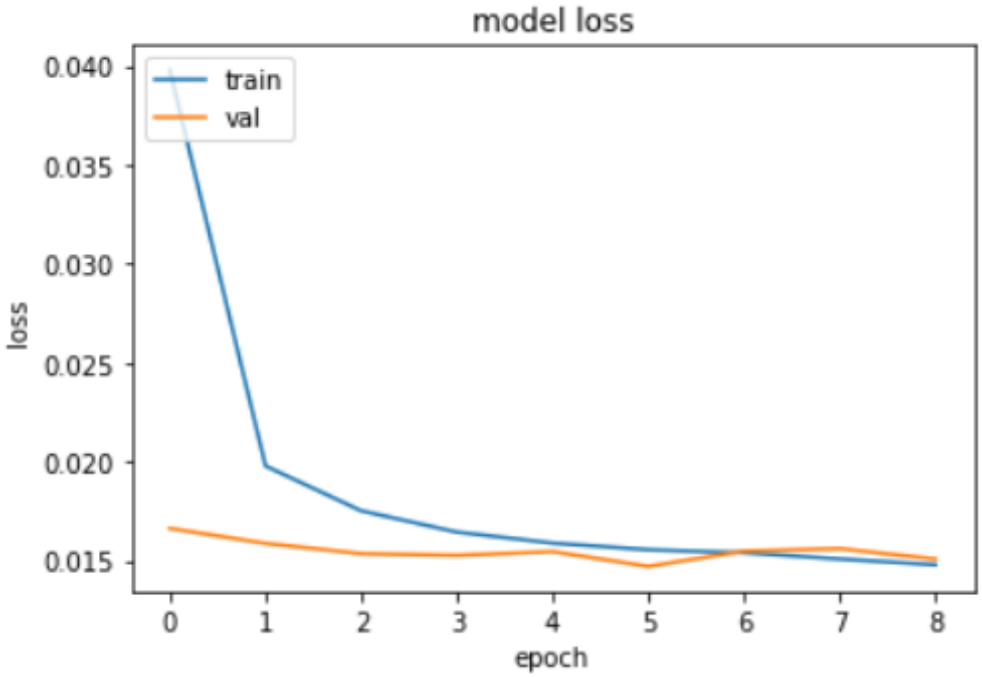

Ilustración 86: Resultado epoch Eugene y Cocoa filtrado 20%

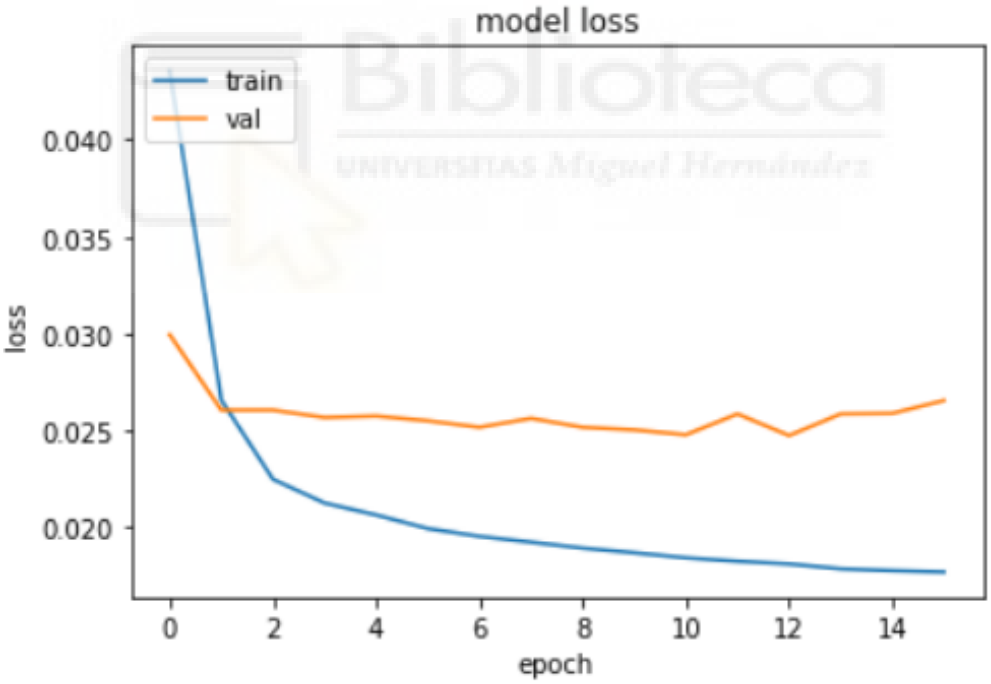

Ilustración 87: Resultado epoch Eugene y Cocoa filtrado 50%

Tras el análisis de los gráficos, podemos comprobar como en el filtrado del 20%, las predicciones obtenidas al entrenar la red neuronal convergen con los valores reales tras realizar 8 iteraciones de entrenamiento. Sin embargo, aunque con el filtrado del 50% se logre minimizar las pérdidas, rápidamente aumentan al

incrementar el número de iteraciones de entrenamiento y, por tanto, obtiene peores resultados.

3.3.5 APLICANDO EL LOGARITMO

Otra de las pruebas que se han realizado en esta investigación es la de aplicarle el logaritmo neperiano a los valores de un parámetro en concreto. En particular, se ha realizado para el parámetro I_s de la placa solar *aSiMicro_03060* en la región de Cocoa. Como los valores de I_s son muy pequeños, el propósito de esto es aplicarles el logaritmo para comprimir toda esta columna de valores. Dicho de otra manera, mantenerlos constantes alrededor de un pequeño intervalo para ver si de esta forma la red neuronal se comporta mejor.

Este proceso se realizará antes de entrenar la red neuronal, por lo tanto, en vez de usar el parámetro I_s como una de las salidas de la red neuronal, usaremos el logaritmo de I_s.

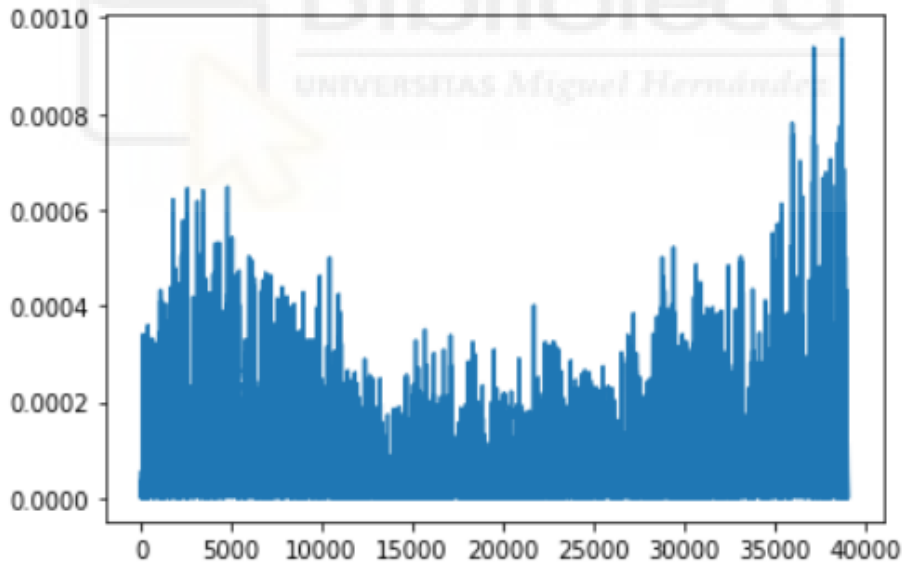

Ilustración 88: Representación de datos de I_s.

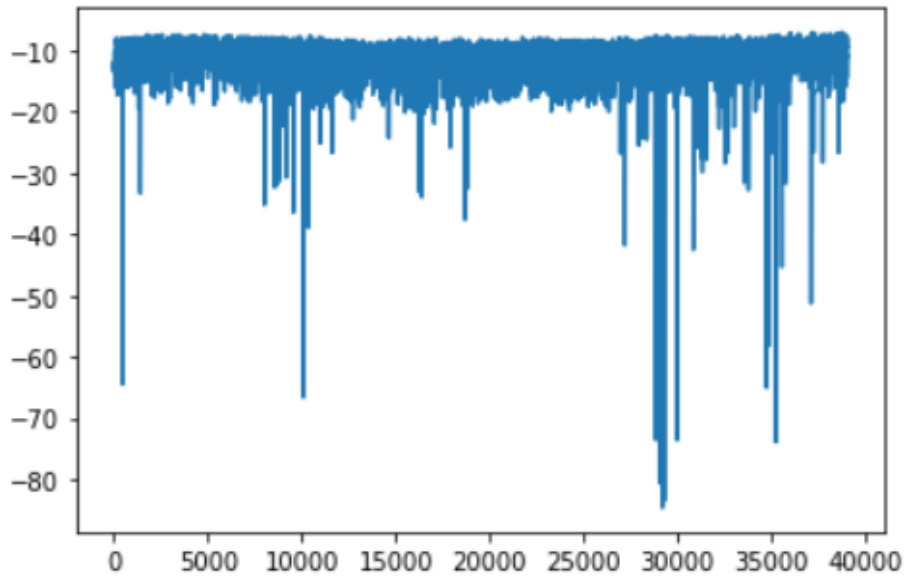

Ilustración 89: Representación de datos Logaritmo de I_s.

Si nos fijamos en las gráficas, se demuestra que aplicando el logaritmo los datos son más lineales, con ciertos valores concretos que distan del resto de datos.

Tras hacer las mismas operaciones, es decir, entrenar la red neuronal, comprobar el error porcentual absoluto medio y la desviación típica, obtenemos los siguientes resultados para el logaritmo del parámetro I_s:

Desviación típica	0.0729
Error porcentual absoluto medio	7.1461%

Tabla 4: Error porcentual absoluto medio y desviación típica aplicando el logaritmo.

No obstante, debemos llevar cuidado y no confundir estos valores con los parámetros I_s, que son los que nos interesen. Para poder comprobar si se ha corregido el error porcentual absoluto medio del I_s tendremos que hacer la inversa al logaritmo, es decir, la exponencial. Aplicaremos la exponencial tanto a sus columnas obtenidas tras entrenar la red neuronal de test, como a las de predicciones.

Por último, con estos datos volvemos a obtener el error porcentual absoluto medio mediante la siguiente expresión:

$$\frac{|\text{Test} - \text{Predicciones}|}{|\text{Test}|}$$

Ilustración 90: Fórmula de error porcentual absoluto medio.

Recordamos los resultados del parámetro Is en el panel solar *aSiMicro_03036* en Cocoa:

Desviación típica	1.379598e+30
Error porcentual absoluto medio	2.3196e+30%

Tabla 5: Desviación típica y Error porcentual absoluto medio de I_s sin preprocesamiento.

A continuación, se muestra los resultados tras entrenar la red neuronal con el logaritmo del parámetro I_s y haber realizado la exponencial para obtener los valores de error y desviación típica de I_s:

Desviación típica	5.971968e+29
Error porcentual absoluto medio	1.01702e+30%

Tabla 6: Desviación típica y Error porcentual absoluto medio de exponencial del parámetro Is

.

Los datos reflejan resultados muy similares y apenas se ven cambios entre antes y después de aplicar el logaritmo. Por esta razón, no vamos a realizar el mismo proceso con otras placas solares ni en otras ciudades.

3.4 TECNOLOGÍA XSI

Cuando hablamos de tecnología xSi nos referimos a paneles solares monocristalinos compuestos por células de silicio procedentes de un único cristal. La fabricación consiste en cortar un lingote de cristal de silicio en láminas. Por lo tanto, su pureza es excelente y esto hace que mejore su eficiencia.

Con el uso de estos paneles podemos conseguir una mayor potencia ocupando poca superficie [23].

3.4.1 RESULTADOS SIN PREPROCESAMIENTO: GOLDEN

Para terminar esta investigación, además de usar otra tecnología como es la de xSi, se ha utilizado la última ciudad de las cuales disponemos medidas, es decir, Golden. Este dataframe está compuesto por una cantidad de filas bastante inferior a las anteriores, conformado por 11929 filas.

Vamos a realizar las mismas pruebas que se han efectuado para la tecnología aSiMicro, pero para xSi. De esta manera, veremos como se comporta otra tecnología diferente en una ubicación distinta a las vistas con anterioridad.

A continuación, como ha sido habitual en este trabajo, vamos a mostrar los percentiles sin aplicarles ningún tipo de preprocesamiento:

	n	Rs	Rsh	Iph	Is
count	11929.000000	1.192900e+04	11929.000000	11929.000000	1.192900e+04
mean	1.364809	5.030577e-03	18.927536	2.460881	8.487796e-07
std	0.079697	3.499244e-03	8.379862	1.740916	1.579754e-06
min	0.053480	6.182072e-08	0.119681	0.168192	8.567741e-197
2%	1.248592	1.709164e-05	7.214318	0.279692	9.405047e-09
5%	1.267464	4.798689e-05	8.508524	0.339153	1.602058e-08
10%	1.287155	1.385868e-04	9.493315	0.452566	2.535036e-08
25%	1.329007	2.840698e-03	11.531534	0.869476	5.446599e-08
50%	1.368167	6.252507e-03	17.176877	1.977732	1.449696e-07
75%	1.403085	7.038492e-03	26.021202	4.149991	8.656650e-07
80%	1.411331	7.153706e-03	27.574343	4.511276	1.279666e-06
98%	1.501279	7.709452e-03	35.756181	5.482084	6.105116e-06
max	2.090581	1.155328e-01	61.084086	6.926911	2.371428e-05

Ilustración 91: Percentiles Golden.

Lo más destacado aquí es el valor mínimo del parámetro I_s, ya que es de 8.567741e-197.

Siguiendo los mismos pasos que se han realizado en los apartados anteriores, vamos a entrenar la red neuronal para, a continuación, averiguar su *Desviación típica* y su *Error porcentual absoluto medio*.

I_s	1.5968
R_{sh}	0.2764
R_s	727.5364
n	0.0208
I_{ph}	0.6317

Tabla 7: Desviación típica Golden sin filtrado.

I_s	125.7338%
R_{sh}	15.6546%
R_s	4900.8014%
n	2.6417%
I_{ph}	36.0750%

Tabla 8: Error porcentual absoluto medio Golden sin filtrado.

Como vemos en las tablas anteriores, la dinámica es similar a las placas aSiMicro. Vemos que los porcentajes más elevados siguen siendo el de los parámetros R_S e I_S. Pero en este caso el parámetro I_S no es tan elevado como pasaba en la placa solar *aSiMicro_03036.*

3.4.2 FILTRADO DE OUTLIERS DATOS: GOLDEN

Acabamos de ver, tanto el error porcentual absoluto medio como la desviación típica sin aplicar ningún tipo de manipulación a los datos. La última prueba que vamos a realizar es el filtrado de datos mediante los gráficos de cajas y bigotes.

Nuevamente, vamos a filtrar:

- El 20% de los datos y posteriormente ver sus resultados.
- El 50% de los datos y posteriormente ver sus resultados.

- **FILTRADO DEL 20%:**

Sin entrar en detalle, vamos a visualizar los diagramas de cajas y bigotes obtenidos en la región de Golden antes de realizar ningún tipo de filtrado. De esta manera, podemos comprobar aquellos valores de nuestra muestra que distan del resto en los cinco parámetros estudiados.

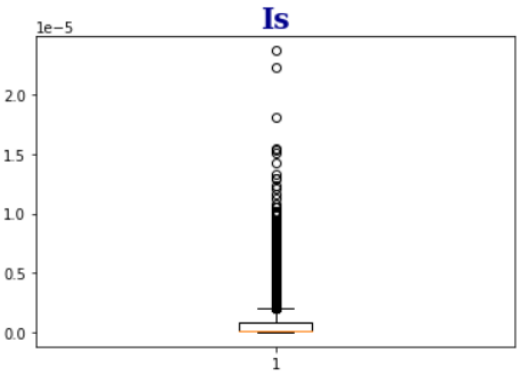

Ilustración 93: Diagrama cajas y bigotes Is Golden sin filtrar.

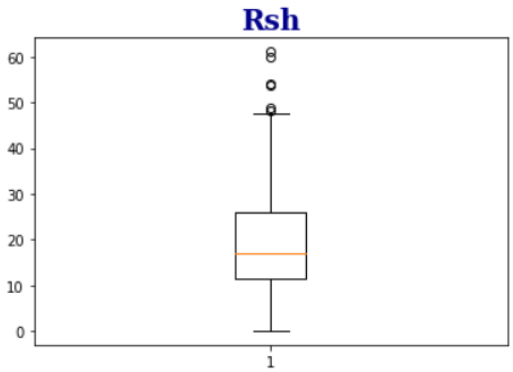

Ilustración 92: Diagrama cajas y bigotes Rsh Golden sin filtrar.

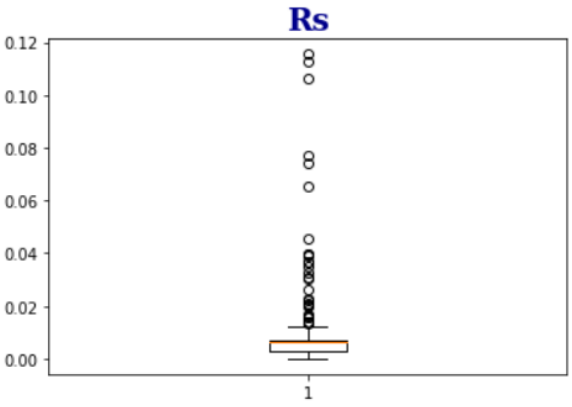

Ilustración 95: Diagrama cajas y bigotes Rs Golden sin filtrar.

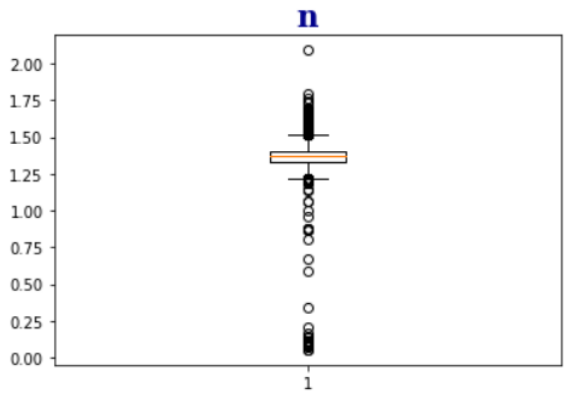

Ilustración 94: Diagrama cajas y bigotes n Golden sin filtrar.

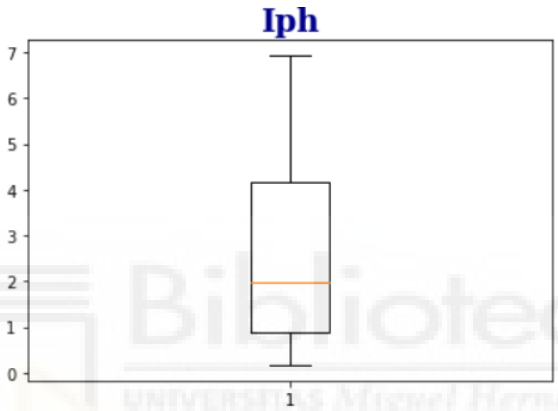

Ilustración 96: Diagrama cajas y bigotes Iph Golden sin filtrar.

Podemos apreciar en las figuras anteriores que la cantidad de outliers que tenemos en este caso con respecto a los que hemos encontrado anteriormente son inferiores. De hecho, para la variable R_{sh} únicamente encontramos cuatro valores atípicos. Esto también era más probable debido a que la cantidad de datos es bastante inferior, recordamos que consta de 11929 filas, frente a los 39037 de Coca, 43343 de Eugene y 82380 de estos dos concatenados.

A pesar de que los resultados en este sentido son mejores, vamos a realizar el filtrado del 20% de los datos eliminando aquellos valores que pueden afectar de manera negativa a nuestra muestra. De esta forma, eliminaremos un total de 2371 filas de nuestro dataframe quedándonos con las 9558 restantes, que dan lugar a los siguientes percentiles:

	n	Rs	Rsh	Iph	Is
count	9558.000000	9558.000000	9558.000000	9558.000000	9.558000e+03
mean	1.359139	0.004848	19.698716	2.171113	3.477900e-07
std	0.055498	0.002531	7.752399	1.516034	4.912057e-07
min	1.213468	0.000042	4.494131	0.189556	5.237462e-10
2%	1.251046	0.000060	7.934387	0.290061	9.655367e-09
5%	1.266616	0.000099	9.136703	0.345493	1.554321e-08
10%	1.283276	0.000268	10.353547	0.461068	2.437565e-08
25%	1.320331	0.002937	12.898377	0.888765	4.934887e-08
50%	1.360960	0.006012	18.702059	1.712568	1.184885e-07
75%	1.395741	0.006831	26.100646	3.391819	4.113431e-07
80%	1.404682	0.006974	27.507298	3.777847	5.762560e-07
98%	1.478926	0.007595	34.933447	5.242252	1.908519e-06
max	1.522743	0.007837	44.424694	6.032300	2.141616e-06

Ilustración 97: Percentiles filtrados 20% Golden.

Como vemos en la tabla anterior, con el filtrado de datos los percentiles se han modificado. De hecho, el valor mínimo para el parámetro I_S ha crecido considerablemente.

A continuación, vamos a ver cómo ha afectado esto a nuestros cinco parámetros mediante los siguientes diagramas de cajas y bigotes:

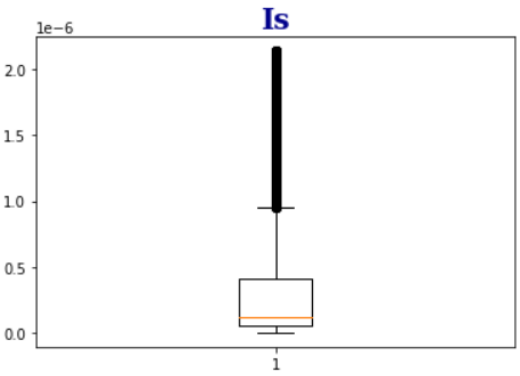

Ilustración 99: Diagrama cajas y bigotes Is Golden 20% filtrado.

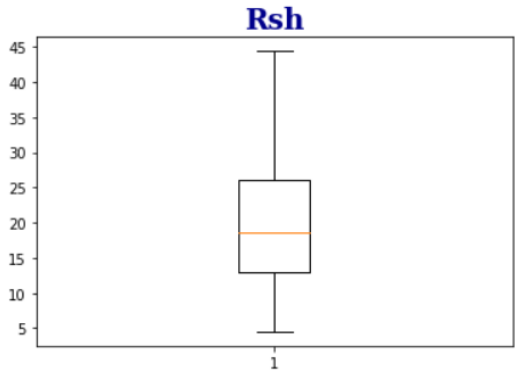

Ilustración 98: Diagrama cajas y bigotes Rsh Golden 20% filtrado.

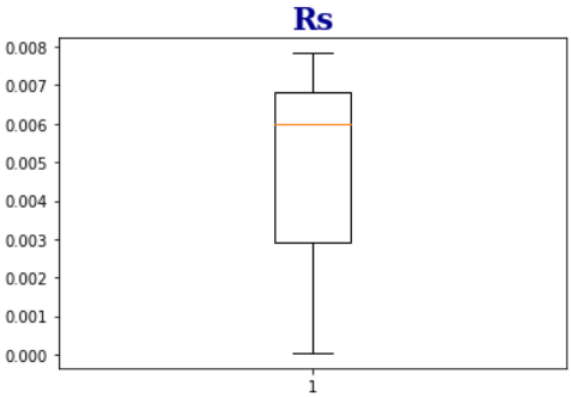

Ilustración 101: Diagrama cajas y bigotes Rs Golden 20% filtrado.

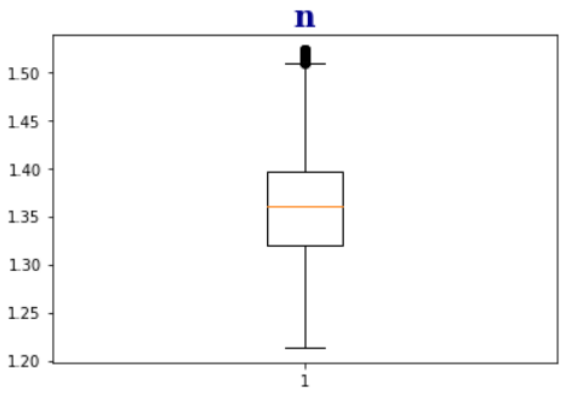

Ilustración 100: Diagrama cajas y bigotes n Golden 20% filtrado.

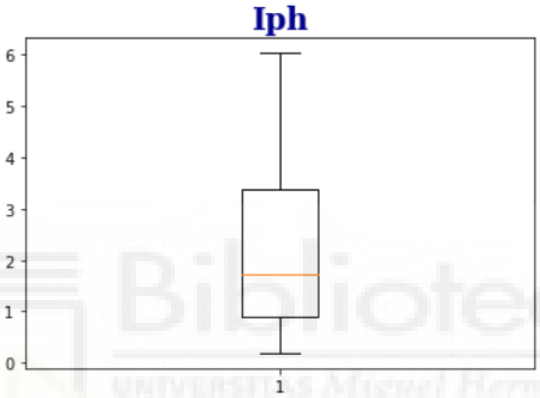

Ilustración 102: Diagrama cajas y bigotes Iph Golden 20% filtrado.

Vemos que con eliminar el 20% de los datos quitar desaparecer casi todos los valores atípicos. Por lo tanto, los resultados del entrenamiento deben haber mejorado.

Veremos, entonces, cuáles son los valores obtenidos de desviación típica y error porcentual tras el entrenamiento de la red neuronal con un filtrado del 20%:

$\mathbf{I_s}$	0.3037
$\mathbf{R_{sh}}$	0.1008
$\mathbf{R_s}$	9.6121
n	0.0178
$\mathbf{I_{ph}}$	0.8605

Tabla 9: Desviación típica Golden filtrado 20%.

$\mathbf{I_s}$	35.8847%

R_{sh}	14.6342%
R_s	326.3332 %
n	2.3863%
I_{ph}	59.2156 %

Tabla 10: Error porcentual absoluto medio Golden filtrado 20%.

Podremos apreciar que se han mejorado tanto los resultados del error porcentual absoluto medio como de la desviación típico. Se debe mencionar que el filtrado se ha realizado basándonos en aquellos parámetros con peores resultados. Es decir, en aquellos cuyo error porcentual era mayor. Aun así, el parámetro R_s continúa teniendo un error porcentual elevado, a pesar de disminuir de 4900% a 326.3332%.

- **FILTRADO DEL 50%:**

Para seguir los mismos pasos realizados con anterioridad, vamos a probar a filtrar el 50% de los datos, eliminando aproximadamente 6000 filas. De esta forma, nuestro dataframe queda con un total de 6021 filas.

A continuación, se muestran los diagramas de cajas y bigotes que nos permiten comprobar que han sido eliminados más outliers que en el caso anterior para cada uno de nuestros cinco parámetros a estudio.

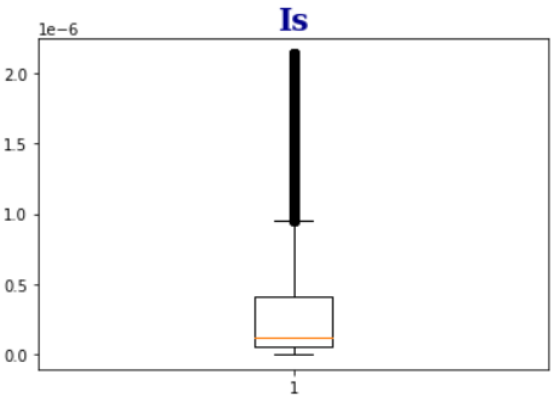

Ilustración 104: Diagrama cajas y bigotes Is Golden 50% filtrado.

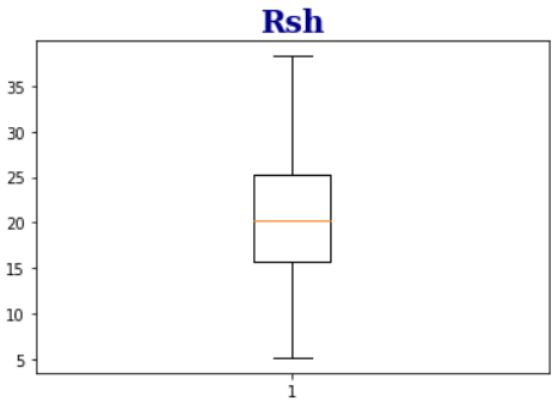

Ilustración 103: Diagrama cajas y bigotes Rsh Golden 50% filtrado.

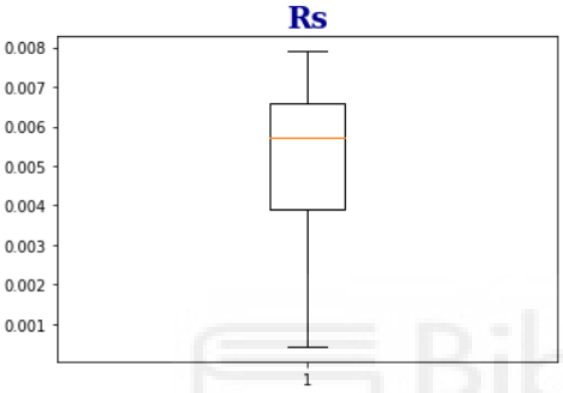

Ilustración 106: Diagrama cajas y bigotes Rs Golden 50% filtrado.

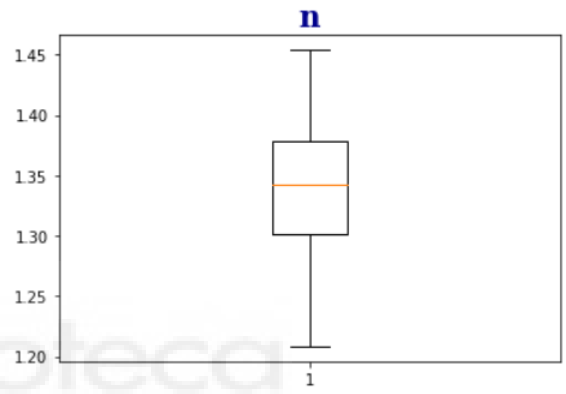

Ilustración 105: Diagrama cajas y bigotes n Golden 50% filtrado.

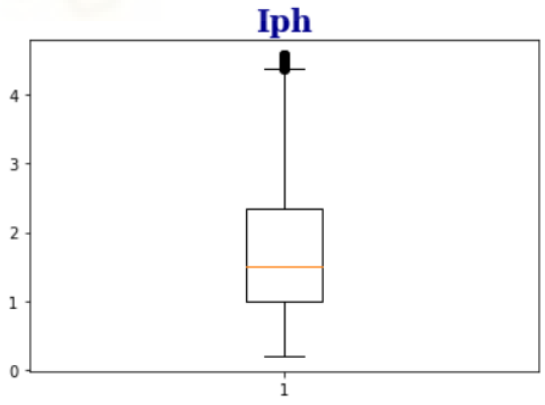

Ilustración 107: Diagrama cajas y bigotes Iph Golden 50% filtrado.

Se puede apreciar que a pesar de que el parámetro Is tiene menos outliers y que en el resto de los parámetros han desaparecido todos los que presentaba, en el parámetro Iph han aparecido nuevos valores atípicos. Esto es debido a que, como se comentó al inicio de la investigación, al eliminar una serie de filas

también se eliminan del resto de parámetros, puede crear la aparición de nuevos outliers.

A continuación, vamos a ver como quedarían los resultados con este filtrado:

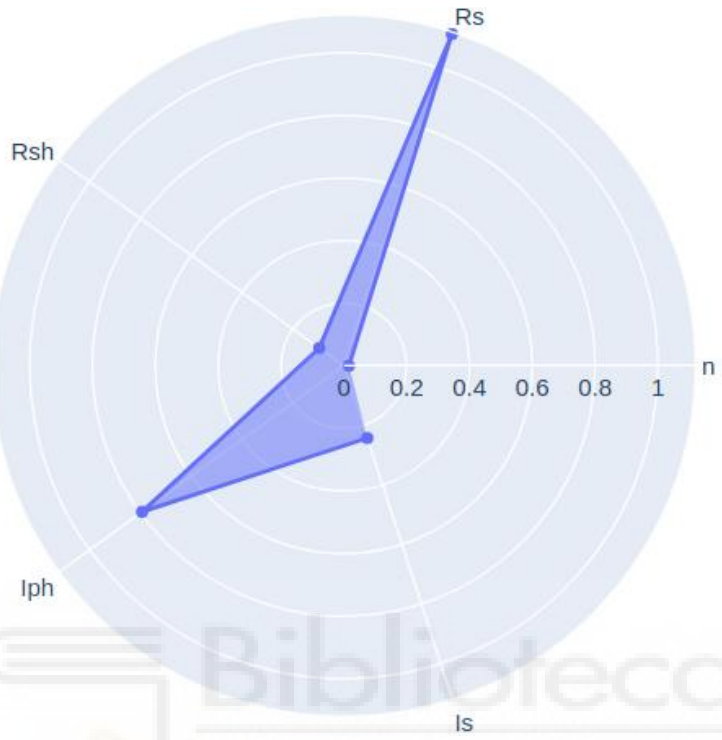

Ilustración 108: Desviación típica Golden filtrado 50%.

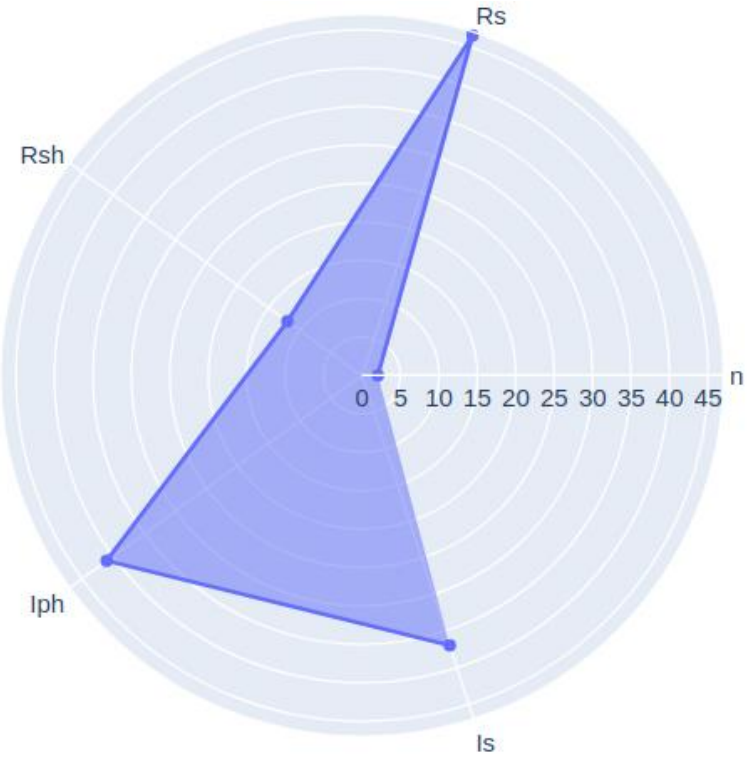

Ilustración 109: Error porcentual absoluto medio Golden filtrado 50%.

Como ya estamos acostumbrados a ver en esta investigación, a medida que se filtran más datos, el error porcentual absoluto medio y la desviación típica decrementan, otorgando aparentemente un mejor resultado. Pero, debemos comprobar de qué manera afecta el entrenamiento de la red neuronal utilizando el modelo de pérdidas que veremos a continuación.

Para ello, comprobaremos los resultados obtenidos a medida que se incrementa el número de epoch para cada uno de los filtrados que hemos realizado. Nos apoyaremos en las gráficas siguientes para hacer un análisis completo:

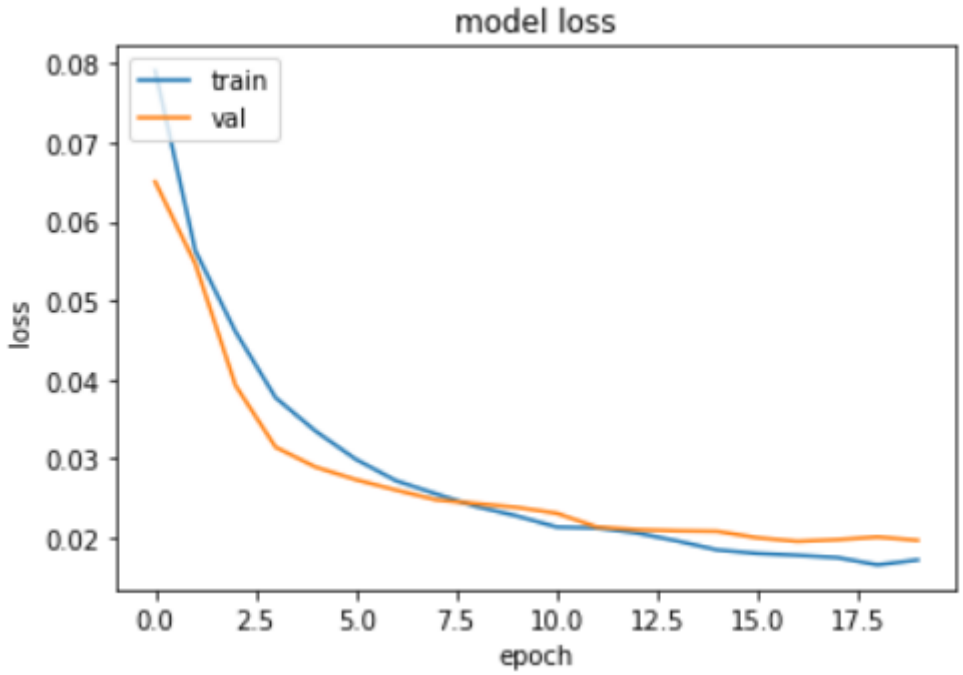

Ilustración 110: Resultado epoch Golden filtrado 20%

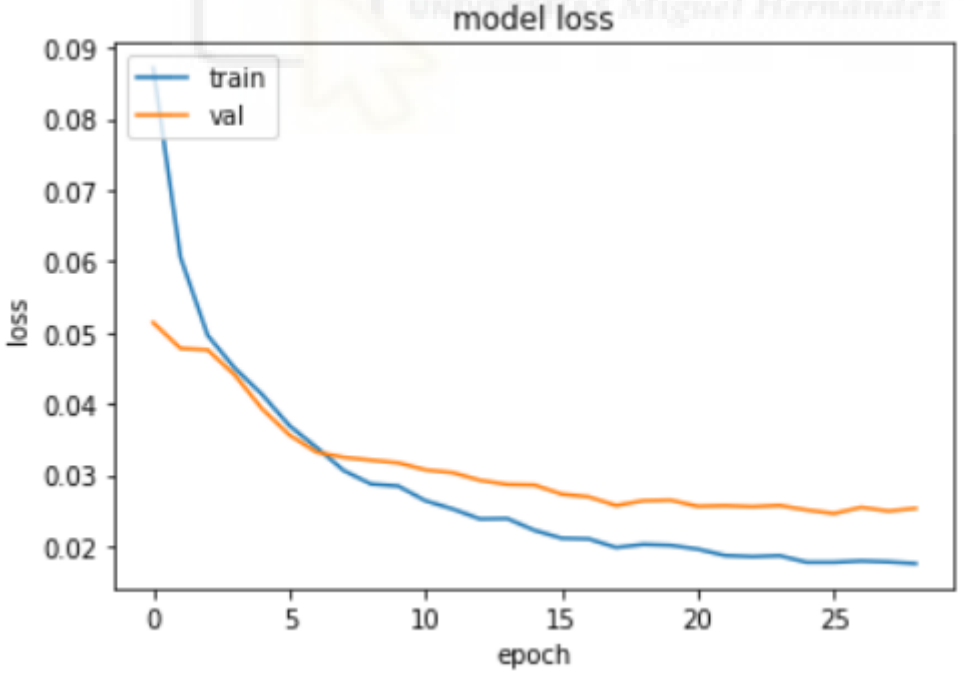

Ilustración 111: Resultado epoch Golden filtrado 50%

Al igual que ocurría en los casos anteriores, un filtrado de datos mayor, como es el del 50%, empeora la distancia entre los valores reales y las predicciones

halladas con el entrenamiento de la red neuronal. Sin embargo, con un filtrado del 20% esta pérdida es minimizada, llegando a converger los valores tomados con los aportados con el entrenamiento.

CAPÍTULO 4 CONCLUSIONES Y FUTURAS INVESTIGACIONES

4.1 CONCLUSIONES

En cuanto a las conclusiones del proyecto podemos decir que en cierta medida se han cumplido los objetivos principales. A lo largo de la investigación se han realizado diferentes pruebas, hemos visto como al entrenar la red neuronal sin realizarle ningún tipo de preprocesamiento obteníamos unas predicciones. Estas tenían en ciertos parámetros un error porcentual absoluto medio y una desviación típica muy elevada.

Tras esto, hemos podido mejorar satisfactoriamente los resultados, realizando un tipo de filtrado de datos. Se ha utilizado el diagrama de cajas y bigotes y posteriormente eliminando percentiles. Seguidamente se ha demostrado que no conviene eliminar una cantidad de datos elevada, llegando a eliminar el 50% de los datos.

Otra de las conclusiones que hemos podido sacar y que se comentará en el siguiente punto, es que el hecho de haber concatenado un mismo modelo de panel solar, pero en diferentes ubicaciones mejora los resultados, sobre todo sin aplicarle ningún tipo de preprocesamiento.

Además, en nuestra prueba realizada de aplicar el logaritmo al parámetro I_s para entrenar la red neuronal de la placa solar aSiMicro_03036 en Cocoa, no ha dado mejores resultados.

En cuanto a los objetivos personales, tanto aprender el lenguaje de programación Python, como profundizar en el campo del Machine Learning y de la Inteligencia Artificial en general estoy contento con los resultados. Puedo

afirmar que el conocimiento que tengo actualmente sobre las mencionadas en relación con cuando empecé el proyecto es muy superior.

4.2 FUTURAS INVESTIGACIONES

En este apartado, se van a incluir algunas de las futuras investigaciones que se pueden seguir haciendo para comprobar si los resultados mejoran.

- Ampliar el tamaño de la muestra: en esta investigación hemos podido hacer la prueba de concatenar los datos obtenidos tras la medición realizada durante un año en Cocoa y Eugene para el panel solar aSiMicro_03036. Lo que se propone hacer estas mismas mediciones para otras ciudades, de modo que la muestra sea mayor tamaño, logrando así una mayor precisión en las estimaciones y, por tanto, un mejor entrenamiento de la red neuronal.
- Modificar filtrados: como se ha visto a lo largo del trabajo, se han realizado filtrados del 20% y del 50%. Para futuras investigaciones, se puede probar a modificar otro tanto por ciento diferente al realizado en esta investigación hasta dar con el idóneo.
- Logaritmo de otros parámetros: hemos realizado para el panel solar aSiMicro_03036 en la ciudad de Cocoa el logaritmo del parámetro I_s antes de entrenar la red neuronal. Se puede realizar esto mismo, pero con otros parámetros, ya sea en este mismo panel y ubicación o en otra.
- Modificar modelo de la Red Neuronal: podría resultar interesante para futuras investigaciones modificar la construcción del modelo de la red neuronal. En vez de poner dos capas ocultas con 50 neuronas cada una, se podría poner una sola capa oculta con más neuronas o crear más capas ocultas con menos neuronas en cada una de estas.

CAPÍTULO 5
BIBLIOGRAFÍA

[1] A. A. C. D. S. G. M. M. G. P. J.-. R. S. K. T. y. T. S. W. Marion, «User's Manual for Data for Validating Models for PV Module Performance,» Golden, 2014.

[2] N. Z. Y. y. B. S. Nahla Mohamed Abd Alrahim Shannan, «Single-Diode Model and Two-Diode Model of PV Modules: A comparasion,» Malasia, 2013, pp. 211-214.

[3] V. G. y. F. J. T. J.M Blanes, «Two-Step Linear Least-Squares Method for Photovoltaic Single-Diode Model Parameters Extraction,» Alicante, 2018.

[4] A. GALIPIENSO, Inteligencia artificial: modelos, técnicas y áreas de aplicación, Paraninfo, 2003.

[5] D. Hinestroza Ramírez, «El Machine Learning a través de los tiempos, y los aportes a la humanidad,» 2018.

[6] J. E. Restrepo, «La mente desencarnada: consideraciones históricas y filosóficas sobre la psicología cognitiva,» de *Psicología desde el Caribe 24*, 2009, pp. 59-90.

[7] J. I. Bagnato, «aprendemachinelearning,» 10 Julio 2018. [En línea]. Available: www.aprendemachinelearning.com.

[8] A. Fernández Khatiboun, «Machine Learning en ciberseguridad,» 2019.

[9] J. Luna Gonzalez, «medium,» 8 Febrero 2018. [En línea]. Available: https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2.

[10] A. Centeno Franco, «Deep Learning,» Sevilla, 2019.

[11] M. G. Arencibia, «DEEP LEARNING: SU IMPACTO ECONÓMICO,» 2020.

[12] D. J. Matich, «Redes Neuronales: Conceptos básicos y aplicaciones,» México, 2001.

[13] F. O. P. R. y. H. F. Castaño, «scielo,» Junio 2007. [En línea]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-33242007000100007.

[14] L. H. y. A. H. Víctor Tomás, «uaeh,» Julio-Diciembre 2011. [En línea]. Available: https://www.uaeh.edu.mx/docencia/P_Presentaciones/huejutla/sistemas/redes_neuronales/introduccion.pdf.

[15] Á. Robledano, «openwebinars.net,» 23 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-python/.

[16] R. Pernaz, «crehana,» 28 Enero 2021. [En línea]. Available: www.crehana.com/es/blog/web/que-es-python/.

[17] R. González Duque, «Python para todos».

[18] E. Díaza García, «Manual de uso de Jupyter Notebook para aplicaciones docentes,» 2018.

[19] L. Toro, «blog.desdelinux.net,» [En línea]. Available: https://blog.desdelinux.net/jupyter-notebook/?utm_content=buffer372ff&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer.

[20] «sintegra,» [En línea]. Available: http://sintegra.es/energias_fotovoltaica.pdf.

[21] E. Cabana, «aprendeconeli,» 13 Septiembre 2020. [En línea]. Available: https://aprendeconeli.com/que-es-un-outlier-atipico/.

[22] «estadisticaparatodos,» [En línea]. Available: https://www.estadisticaparatodos.es/taller/graficas/cajas.html.

[23] «artesolarfotovoltaica,» 10 Marzo 2020. [En línea]. Available: https://www.artesolarfotovoltaica.com/nuevos-paneles-de-celula-partida/.