UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

OPTIMIZACIÓN SUBROGADA DE UNA PLANTA SOLAR TÉRMICA INTEGRADA A PROCESOS INDUSTRIALES MEDIANTE HERRAMIENTAS DE MACHINE LEARNING

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL MECÁNICA

ANA VALENTINA PUENTES ALARCÓN

PROFESOR GUÍA:
FRANK DINTER

MIEMBROS DE LA COMISIÓN:
FRANCISCO FUENTES HERLITZ
VIVIANA MERUANE NARANJO

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL MECÁNICA
POR: ANA VALENTINA PUENTES ALARCÓN
FECHA: 2023
PROF. GUÍA: FRANK DINTER

OPTIMIZACIÓN SUBROGADA DE UNA PLANTA SOLAR TÉRMICA INTEGRADA A PROCESOS INDUSTRIALES MEDIANTE HERRAMIENTAS DE MACHINE LEARNING

Considerando el gran recurso solar que posee Chile, la generación de energía térmica mediante energía solar es una propuesta con gran potencial, pero es necesario que estas sean atractivas en un nivel tecno-económico. Una forma de lograr esto es mediante el uso de algoritmos computacionales, así el objetivo general del presente trabajo corresponde a desarrollar y aplicar un modelo basado en algoritmos de Machine Learning para optimizar el diseño de una planta solar térmica integradas a procesos industriales.

La metodología sería diseñar los modelos de Machine Learning mediante herramientas de calibración (tuner) de hiperparámetros, identificar que tipo de preprocesamiento tiene menor error para ser utilizados en la optimización, diseñar los algoritmos de optimización para finalmente realizar nuevas simulaciones para la validación de resultados.

Para esto se utilizaron Redes Neuronales y Árboles de Decisión entrenados con dos datasets diferentes, uno uniforme con datos equiespaciados, y el otro con datos al azar, dentro de respectivos rangos. También se utilizaron preprocesamientos diferentes para analizar cual permite un mejor aprendizaje de los modelos. Se realizaron optimizaciones correspondientes mediante Algoritmos Genéticos y su variación NSGA-II. Todos los modelos fueron desarrollados en el lenguaje de programación Python utilizando las librerías TensorFlow, Scikit-Learn, PyGad y Pymoo principalmente.

Los resultados obtenidos corresponden a que el mejor tipo de preprocesamiento para todos los modelos correspondió a Standard Scaler, donde se obtuvieron los menores errores de predicción; para el caso de Redes Neuronales entrenados con el primer dataset mencionado un 0.12 %. También que para este caso Redes Neuronales tuvo un mejor desempeño de Árboles de Decisión, y que los modelos entrenados con el dataset uniforme era capaz de entregar mejores resultados con una gran capacidad de predicción e interpolación de los datos.

Así, en el trabajo se concluye que la optimización multiobjetivo con el modelo de Redes Neuronales entrenados con un dataset uniforme entrega resultados con muy bajo error porcentual, bajo el 2 %, en comparación a los nuevamente simulados. También se concluye que los modelos de Machine Learning utilizados en la optimización deben tener un muy bajo error, para evitar problemas en la optimización y que los resultados no sean concluyentes.

Para mi familia
Los amo

Agradecimientos

En primer lugar, quiero agradecer a mi familia por acompañarme en todo momento, y estar juntos en las buenas, malas y mas o menos. Gracias a todos por ser una familia tan unida, con tanto amor para entregar y repartir, por enseñarme tantas cosas lindas de la vida, y regalarme tantas memorias felices. En particular a mi mamá, Paulina, por acompañarme con tanto amor y cariño en este proceso y en toda mi vida, por enseñarme y guiarme en cada situación, gracias por siempre confiar en mi. A mis tías y tíos, que me entregan cariño, ayuda y apoyo, gracias por estar siempre presentes. A mi hermano y primos que son como mis hermanos, los adoro.

A Pamela, Daniela, Josefa, Sofía, Francisca, Romina, Consuelo, Franco, Matías, Luís, Cristóbal y José. Gracias por acompañarme en tantos procesos y en diferentes etapas de la vida. Cada uno de ustedes me ha entregado cariño, aprendizajes y memorias atesoradas. Tantas risas, alegrías, noches de juegos y salidas. Gracias por ser parte de mi vida, y ser los mejores.

A Hugo por estos años de tanto amor y comprensión. Gracias por acompañarme, y ser un apoyo tan importante. No tengo palabras para expresar lo lindo e importante que ha sido para mi todo lo que nos hemos enseñado, aprendido y querido.

A mis profesores guía y co-guía. Gracias por su apoyo, comprensión y ayuda en este proceso. Gracias por las herramientas y aprendizajes a lo largo de este trabajo.

Finalmente, quiero agradecer a todas las personas que alguna vez fueron parte de mi vida y me regalaron memorias lindas. Los recuerdo y llevo en el corazón por siempre.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

A nivel mundial uno de los principales problemas al que la humanidad debe enfrentar es la contaminación. Una de las fuentes más grandes de contaminación corresponde a la quema de combustibles para la generación de energía, sea esta térmica, eléctrica, entre otros. Estos generan emisiones de dióxido de carbono y gases invernadero que impactan altamente en el calentamiento global; en particular, la generación de energía térmica para procesos industriales es una de las principales fuentes de contaminación.

En Chile, según el balance nacional de energía del año 2020 [1], un 40 % de la total producida en el año correspondía a energía utilizada en industrias. Dentro de ese porcentaje un 63 % corresponde a energía térmica generada por la quema de combustibles que generan gases invernadero y contribuyen a la contaminación. Es por esto que el gobierno de Chile tiene proyectos de descarbonización del país, donde se busca alcanzar la carbono-neutralidad y la neutralidad de gases de efecto invernadero para el año 2050. Teniendo esto en cuenta, es necesario formular soluciones atractivas y rentables que permitan ese objetivo, y en particular encontrar soluciones para generar energía térmica capaz de satisfacer la demanda asociada a procesos industriales.

Chile es reconocido como uno de los países con mayor recurso solar, con una irradiación global horizontal (GHI) y directa normal (DNI) promedio de 5,33 y 6,96 [kWh/m^2/día] respectivamente [2]. Es decir, el país cuenta con un alto potencial para la implementación de tecnologías de captación solar. Considerando que el rango de temperaturas en procesos industriales abordables con energía solar térmica es de 350ºC [3], y los procesos en Chile en su mayoría no superan los 400ºC [4], la utilización de plantas solares térmicas es una solución viable para satisfacer la demanda energética. Así, en este trabajo de título se busca contribuir al proceso de descarbonización y reducir las emisiones de gases invernadero optimizando el diseño de plantas solares térmicas; para reducir los tiempos de diseño, como también para lograr que este tipo de tecnología sea atractiva a nivel tecno-económico.

Considerando la magnitud de los proyectos de plantas solares térmicas, la mejor manera de realizar este proceso de optimización es utilizando herramientas computacionales avanzadas. Utilizando algoritmos que permitan que este proceso sea automático, como también reducir

los tiempos de optimización y en consecuencia de diseño. Es decir, se busca evaluar una metodología y un modelo de Machine Learning que permita la optimización de plantas solares térmicas integradas a procesos industriales.

1.1. Objetivos

Con lo mencionado anteriormente, a continuación se detallan los objetivos del trabajo, tanto como el objetivo general, como los específicos a lograr a lo largo del desarrollo del trabajo.

1.1.1. Objetivo general

Desarrollar y aplicar un modelo basado en algoritmos de Machine Learning para optimizar el diseño de una planta solar térmica integrada a procesos industriales.

1.1.2. Objetivos específicos

- Analizar el problema y plantear el modelo de optimización, con sus variables respectivas, restricciones y finalmente la función objetivo que se busca optimizar.
- Aplicar modelos de algoritmos capaces de aprender y predecir la producción de las plantas solares térmicas.
- Comparar dos modelos de Machine Learning aplicados al problema y técnicas de pre-procesamientos de datos.
- Aplicar algoritmos de optimización capaces de resolver el problema planteado.
- Evaluar la metodología planteada basado en los resultados obtenidos, y realizar la validación de esta con nuevas simulaciones.

1.2. Alcances

Es importante mencionar también los alcances del trabajo, es decir, hasta que punto abarca el trabajo y que cosas no contempla.

- El modelo de la planta solar es hecho y entregado por la empresa Centro Fraunhofer Chile Research, como también las bases de datos con las configuraciones y energías que esta entrega.
- El trabajo realizado queda restringido a un proceso industrial en particular según la demanda energética que este requiera.
- La planta solar térmica utilizada queda restringida a una locación en particular. Es decir, se está optimizando una planta solar en específico según los datos que fueron entregados.

Capítulo 2

Antecedentes

En este capítulo se exponen y explican los diferentes temas relevantes para el estudio realizado. Es decir, la energía solar térmica integrada a la industria, su aplicación en Chile, y las herramientas de Machine Learning utilizadas. Como también la revisión bibliográfica utilizada para el desarrollo del trabajo.

2.1. Energía solar térmica integrada a la industria

Con la tecnología adecuada, la energía solar térmica se puede utilizar para cubrir la demanda de calor en procesos de calor y frío en diversos procesos de la industria. Los sistemas solares térmicos tienen una mayor eficiencia para procesos industriales de bajas temperaturas, pudiendo ser utilizada, por ejemplo, en la industria textil para para procesos de lavados o teñido, en la industria láctea para procesos de pasteurización, como también en la industria minera para lixiviación [5].

Integrar calor solar a procesos industriales es más complejo que el uso de tecnologías convencionales. Para realizar un sistema eficiente es necesario considerar múltiples variables para alcanzar un óptimo tanto técnico como económico, capaz de satisfacer la demanda energética como el nivel de temperatura, teniendo en consideración que en general los sistemas convencionales están sobredimensionados. Esta medición cuantitativa tecno-económica para plantas solares térmicas se realiza mediante el parámetro LCoH, que corresponde al "Levelized Cost of Heat", es decir, cuantifica el costo por cada unidad de energía generada. Este valor se calcula mediante la ecuación 2.1 [4].

$$LCoH = \frac{I_0 + \sum_{t=1}^{T} \frac{C_t}{(1+r)^t}}{\sum_{t=1}^{T} \frac{E_t}{(1+r)^t}} \tag{2.1}$$

Donde I_0 corresponde a la inversión inicial, C_t a los costos de operación y mantenimiento, E_t a la demanda energética y r a la tasa de descuento.

2.1.1. Proceso de integración y sistema solar térmico

La integración de la energía solar a procesos industriales es posible realizarla en diferentes puntos del suministro de calor, es decir, de la planta solar y de la planta industrial. Para identificar estos puntos es necesario tener en consideración el proceso industrial, la especificación de los conceptos de integración, planificar el proceso del sistema térmico solar y el criterio de selección de puntos de integración mas prometedores [6].

Para seleccionar el sistema de integración más apropiado, en primer lugar es necesario identificar la categoría del proceso industrial. Estas son, calentamiento de un fluido de trabajo, mantención o aumento de temperatura de tanques de almacenamiento y procesos de separación térmica. Esta clasificación permite desarrollar integraciones que sean apropiadas para cada una de estas.

Las plantas solares térmicas para procesos industriales se componen de colectores solares, los cuales transforman la radiación solar a energía interna de algún fluido de transporte, es decir, aumentan su temperatura. Es importante mencionar que según la demanda térmica a satisfacer, se utilizan distintos tipos de colectores. Los colectores de placa plana se utilizan para temperaturas de 30°C a 80°C; los colectores de tubos entre 50°C a 130°C; los reflectores fresnel lineal entre 60°C a 400°C; y por último los colectores cilindro-parabólicos de 100°C a 450°C. En general las plantas también tienen controladores, es decir, un sistema de regulación para controlar las temperaturas y las bombas de impulsión para los fluidos de transporte. Por último, estas también poseen intercambiadores de calor, las cuales permiten aumentar la temperatura del fluido utilizado en el proceso industrial, o generación de vapor según corresponda. Algunas plantas también poseen tanques de almacenamiento, dependiendo de las condiciones del sistema térmico [7].

2.2. Industria Chilena e integración de energía solar

En la industria Chilena, los procesos industriales en su mayoría corresponden a procesos de bajas a medias temperaturas, es decir entre los 100°C hasta los 400°C, lo que significa que gran parte del calor de industria Chilena podría ser suplido por energía solar térmica. Una de las plantas más famosas ya existentes en el país corresponde a la planta solar integrada a procesos de electro-obtención en la división Gabriela Mistral de Codelco en la región de Antofagasta. Esta planta aporta 54 GWh/año aproximadamente, lo que equivale al 80 % de energía por combustible fósil utilizada anteriormente [8].

Además de las bajas temperaturas promedio que requieren los procesos de la industria Chilena, es muy importante destacar el gran recurso solar que se tiene en el país, es por esto que las temperaturas mencionadas anteriormente pueden ser suplidas por energía solar térmica. La irradiación solar directa se puede observar en la Figura 2.1. Es importante mencionar que a pesar de que Chile cuenta con uno de los recursos solares más altos del mundo [9], actualmente la implementación de la energía solar tiene muy altos costos, como también

es difícil alcanzar un diseño tecno-económico óptimo.

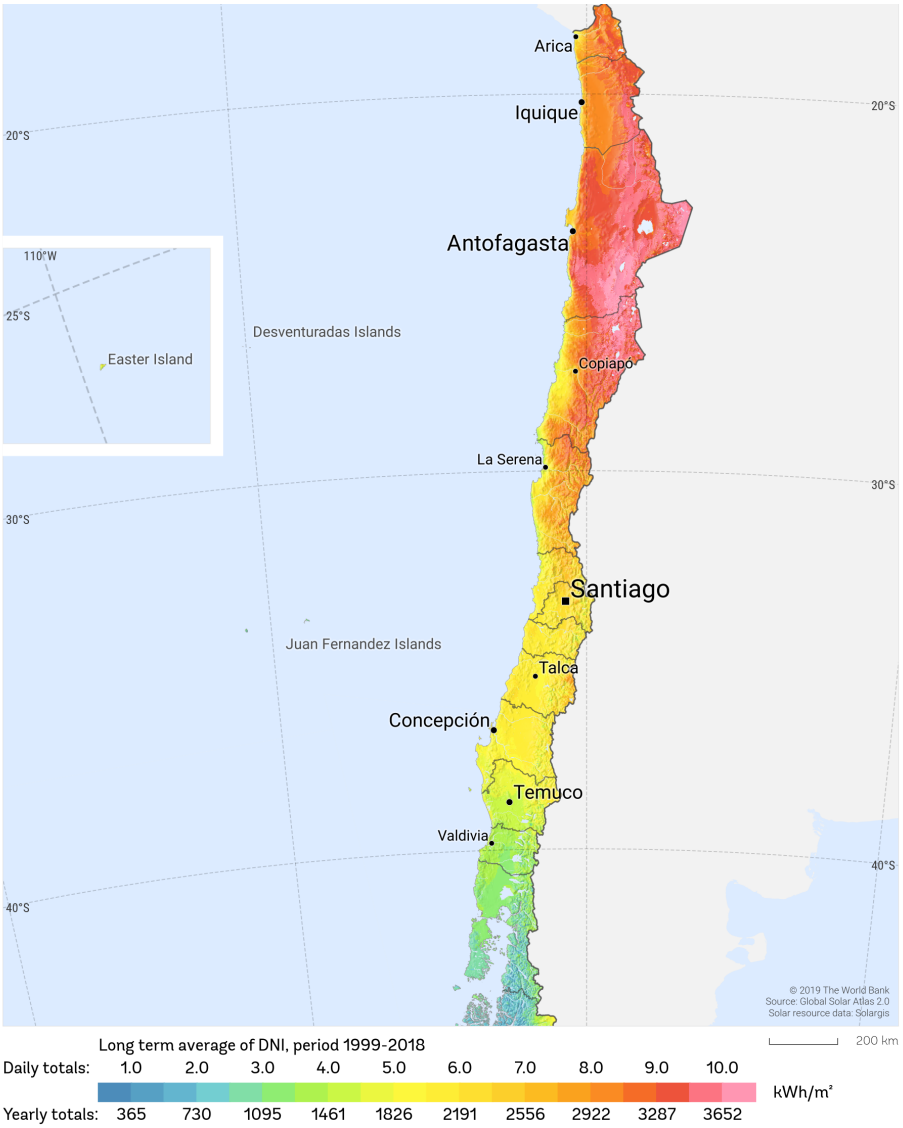


Figura 2.1: Irradiación solar en Chile [10]

Se puede notar que en la zona norte y centro del país la irradiación es bastante alta. En particular el norte de Chile corresponde a la zona que recibe mayor irradiación solar a nivel mundial; gracias a esto, la temperatura promedio nacional no baja de los 20°C anualmente. Por estas dos cosas, la integración de energía solar térmica a procesos industriales es una propuesta realista y eficaz para disminuir la contaminación y satisfacer la demanda térmica, pero con la desventaja que implican los costos asociados a este tipo de integración y el cambio de tecnologías convencionales a tecnología solar. Es por esto que es importante optimizar las plantas solares, para obtener proyectos rentables.

2.3. Algoritmos de Machine Learning y optimización

Dentro de los diferentes lenguajes de programación, existen diversas librerías y algoritmos de Machine Learning que permiten la regresión o clasificación de datos, como también algoritmos que permiten optimizar. En este trabajo se utilizará el lenguaje ***"Python"*** en el cual existen una gran variedad de librerías que permiten realizar las tareas requeridas, y en general no requieren grandes recursos computacionales, permitiendo ser utilizados para diferentes tipos de proyectos y modelos de diversas complejidades.

Como herramienta de Machine Learning para el aprendizaje de los outputs de la planta solar a continuación se presentan lo que son las redes neuronales artificiales o *Artificial Neural Networks* (ANN), escogidos por su precisión [11] y Árboles de Desición o *Decision Trees* (DT), escogidos por los bajos recursos computacionales que requiere, ambos algoritmos de aprendizaje supervisado. Y como herramientas de optimización los Algoritmos Genéticos o *Genetic Algorithms* (GA), que corresponde a un algoritmo que encuentra el óptimo mediante la evaluación de varias posibles soluciones al problema estudiado.

2.3.1. Preprocesamientos para Machine Learning

El preprocesamiento de los datos para el entrenamiento de modelos de Machine Learning es un paso esencial [12] debido a que permiten que la predicción de los modelos se entrenen de mejor manera y así aumentar su capacidad de predicción. El preprocesamiento de datos corresponde a limpiar los datos, integrarlos, transformarlos y reducirlos antes de la siguiente etapa de entrenamiento [13].

En particular, la transformación de los datos, por ejemplo la normalización, se podrían considerar como de uso obligatorio y corresponde a transformar las variables independientes, es decir los inputs del problema. Unos ejemplos se pueden observar en las ecuaciones 2.2 a 2.4 obtenidas de scikit-learn [14].

"MinMax Scaler" (MMS) es una normalización que resta al valor el menor de la lista y lo divide por la diferencia entre máximo y el mínimo, y los deja dentro del rango [0, 1].

$$x_{MMS} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

"MaxAbs Scaler" (MAS) normaliza todos inputs dividiendo por el mayor dejándolos dentro del rango [0, 1].

$$x_{MAS} = \frac{x}{x_{max}} \tag{2.3}$$

Y por último *"Standard Scaler"* (SS) normaliza los datos según su desviación estándar (s) y promedio (μ) dejándolos dentro del rango [-1, 1].

$$x_{SS} = \frac{x - \mu}{s} \tag{2.4}$$

También existen las técnicas de reducción de datos, que corresponden a obtener una representación reducida de los datos originales [13], lo que puede disminuir el tiempo de aplicación de los modelos de Machine Learning. Un ejemplo de esta técnica corresponde a "Principal Component Analysis" (PCA), la cual disminuye las dimensiones de los inputs identificando la información importante mediante combinaciones lineales de los datos originales y así generar un nuevo set de variables ortogonales entre sí con la información "comprimida" [15].

2.3.2. Redes Neuronales Artificiales

Las redes neuronales son un tipo de algoritmo de Machine Learning bio-inspirado en las celulas cerebrales, las neuronas.

La estructura de este algoritmo en términos simples corresponden a capas con diversas neuronas (*nodos*), donde hay una capa de entrada (*input layer*) con una cantidad de nodos igual a la cantidad de datos de entrada, es decir las variables independientes; capas ocultas (*hidden layers*) que pueden tener diversas cantidades de nodos; y por último la capa de salida (*output layer*) que serían los resultados obtenidos, es decir las variables dependientes. En este trabajo se utilizaron Redes con capas densas, donde todos los nodos de una capa están conectados a los nodos de la siguiente. En general la cantidad de capas ocultas y sus cantidades de nodos dependen del problema y la cantidad de datos con los cuales se está trabajando. En la Figura 2.2 se puede observar un diagrama de la estructura de una ANN [16].

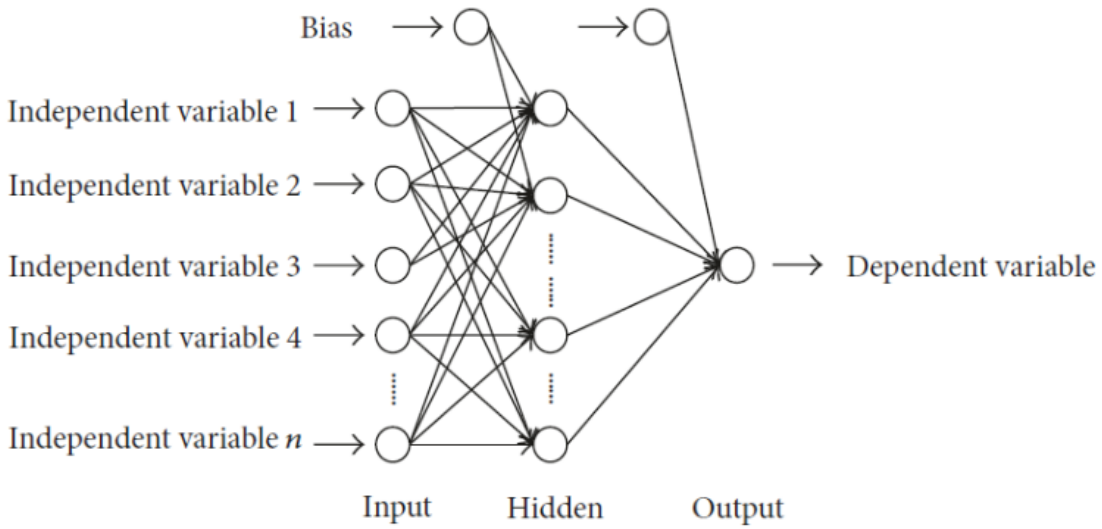


Figura 2.2: Esquema de una red neuronal genérica [16]

El funcionamiento general de este algoritmo se basa en la ponderación de los valores de los diferentes nodos para generar el valor del nodo al que está conectado. Si se tiene una red

densa los valores de los nodos se van construyendo a partir de los pesos de todos los nodos anteriores. Cada nodo además posee una función de activación que determina si el nodo se activa o no.

El entrenamiento de una Red Neuronal requiere una base de datos con los inputs y outputs correspondientes. Estos se exponen a la Red, y esta va aprendiendo mediante el ajuste de los pesos entre nodos. En general la base de datos se divide en tres secciones, el set de entrenamiento, que como dice el nombre es para entrenar la red, el set de validación, utilizado para analizar la capacidad de predicción de la red a medida que esta se entrena, y por último el set de prueba, que permite visualizar la precisión del aprendizaje de la red. Es importante que se utilicen buenos porcentajes de la base de datos para entrenamiento y validación, como también tener una cantidad de capas ocultas con cantidad de nodos óptimos para evitar el under- o overfitting, es decir, que no sea capaz de generalizar de buena forma los resultados no pertenecientes al set de entrenamiento.

2.3.3. Árboles de Decisión

Los árboles de decisión son modelos de Machine Learning eficientes tanto para problemas de regresión y clasificación, y como dice su nombre, se basan en la forma de un árbol para obtener resultados [17], como se puede observar en la figura 2.3 adaptado de [18].

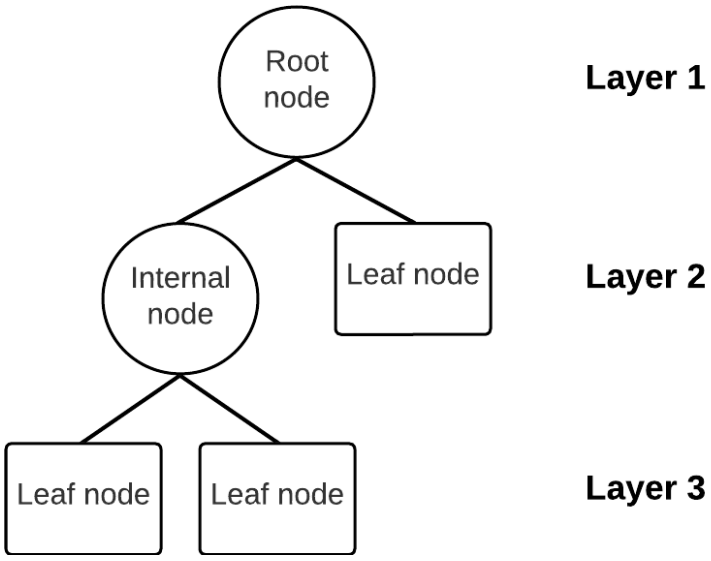


Figura 2.3: Esquema de un Árbol de Desición [18]

Los árboles de decisión se componen del nodo raíz (*root node*) que contiene todos los datos, nodos internos (*internal node*) y los nodos terminales (*leaves*) que finalmente entregan los resultados. Su funcionamiento se basa en que cada nodo se particiona en los nodos siguientes u hojas por recursión [18]. Así, el concepto básico de este modelo corresponde a realizar decisiones de particionar hasta obtener la solución [17].

Para este algoritmo sólo es necesario particionar el set de datos completo en dos conjuntos, el de entrenamiento y finalmente el de testeo. Esta diferencia con el modelo anterior ocurre debido a que en general los algoritmos de las diversas librerías de *"Python"*, en particular *Scikit-Learn*, realizan validaciones cruzadas de forma interna, que sirve para ir validando los datos a medida que el modelo entrena, obteniendo de forma automática un conjunto de testeo para ir probando el desempeño del árbol, símil al conjunto de validación.

2.3.4. Algoritmos Genéticos

Los algoritmos genéticos son algoritmos de evolución biológica inspirados en la teoría evolutiva de Darwin, el cual es útil tanto para la optimización como para modelar sistemas evolutivos [19].

Este algoritmo trabaja sobre candidatos de posibles soluciones del problema a resolver, los cuales se llaman ***individuos***. Cada uno de estos tiene asociada una función ***fitness***, la cuál identifica que tan apto es el individuo dentro de la población. Así, a mayor fitness mejor es la solución al problema, es por esto que la función fitness está directamente ligada a la función objetivo del problema a resolver. Luego, la evolución de estos individuos y de las siguientes generaciones se basan en tres conceptos de la teoría evolutiva, la selección natural, crossover de los genes y la mutación de estos.

La ***selección*** de padres es una función que determina cuales son los individuos que pasarán sus genes a la siguiente generación, donde en general corresponden a los individuos con mayor fitness. La función ***crossover*** permite la combinación genética de dos o más individuos, es decir, es una función que combina los genes de los padres seleccionados para armar los nuevos individuos de la siguiente generación. Por último, la ***mutación*** es otra función que se encarga de cambiar uno o más genes al azar de algún individuo según una cierta probabilidad con el fin de dar más variabilidad a los individuos. Esto último mencionado es de suma importancia, ya que al dar variabilidad al problema se permite que los resultados no se estanquen en un óptimo local [20].

Existen diversas aplicaciones de Algoritmos Genéticos dependiendo del problema a resolver. En particular existen modelos diferentes para trabajar con un objetivo y multi-objetivos.

La optimización de un objetivo utilizando algoritmos genéticos aplica lo mencionado anteriormente en ciclos de determinadas generaciones. Así, en cada paso del modelo, mientras avanzan las generaciones, lo que se busca es encontrar al individuo que solucione el problema. Este tipo de optimización en general es la forma más sencilla de aplicar los algoritmos genéticos.

En la figura 2.4 adaptada de [21] se puede observar el ciclo correspondiente a los algoritmos genéticos para un objetivo. Donde en primer lugar se crea la población inicial de soluciones al problema, con la cantidad deseada de individuos por generación y los genes dentro de sus rangos correspondientes, luego a cada uno de ellos se les calcula la función fitness. Por último

se seleccionan los padres, se realiza el crossover y mutación definidos y se obtienen los hijos, es decir los individuos de la siguiente generación.

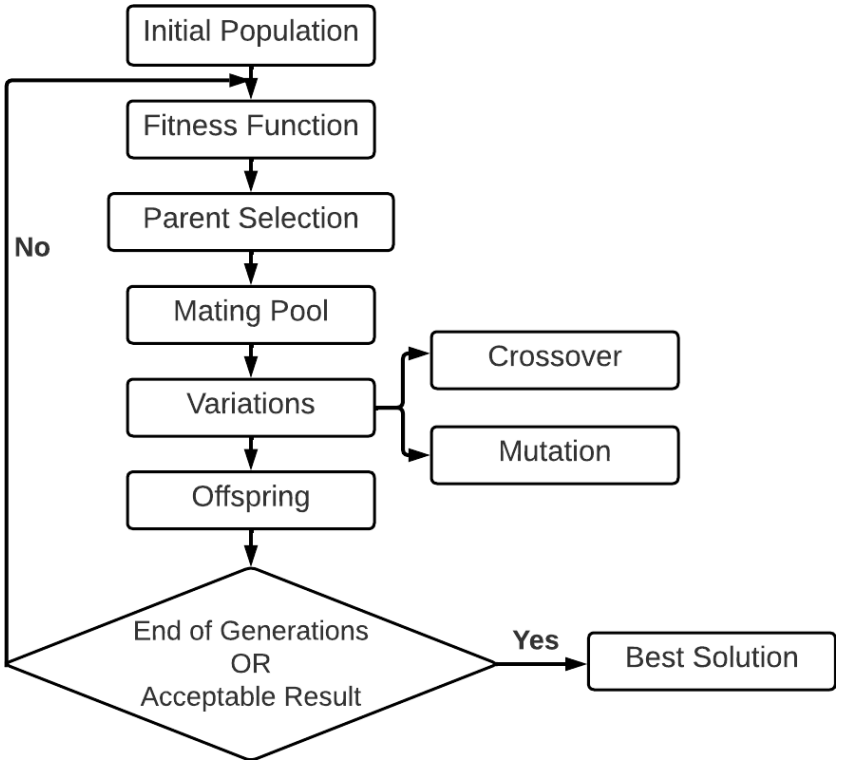


Figura 2.4: Esquema de modelo Algoritmos Genéticos [21]

Para el caso de las aplicaciones multi-objetivo, en la actualidad existen diversos modelos basados en el mismo principio explicado anteriormente, pero agregando funcionalidades nuevas. Uno de estos corresponde a el modelo Nondominated Sorting Genetic Algorithm-II (NSGA-II). El cual permite encontrar las fronteras de Pareto de los objetivos [22].

El funcionamiento de este consiste en crear una población inicial de padres (P_0) que se asigna su respectivo fitness y se ordenan según su no-dominancia. Una solución no-dominada significa que al mejorar un valor de una de las funciones objetivo, empeora uno o mas de los otros objetivos. Estas soluciones se denominan frente óptimo de Pareto.

Luego, con P_0, se crea la siguiente generación (Q_0) mediante selección, crossover y mutación de tamaño N. Ambos conjuntos P_0 y Q_0 se unen y ordenan según fitness de cada uno de los conjuntos de Pareto (F_n). Como este corresponde a un algoritmo elitista, los mejores conjuntos de pareto son escogidos hasta completar la generación de tamaño N, el resto de soluciones son desechados, creando así la siguiente generación (P_1). Lo mencionado se puede observar en la figura 2.5, donde se muestra el loop del algoritmo para la generación t perteneciente a la cantidad de generaciones a estudiar.

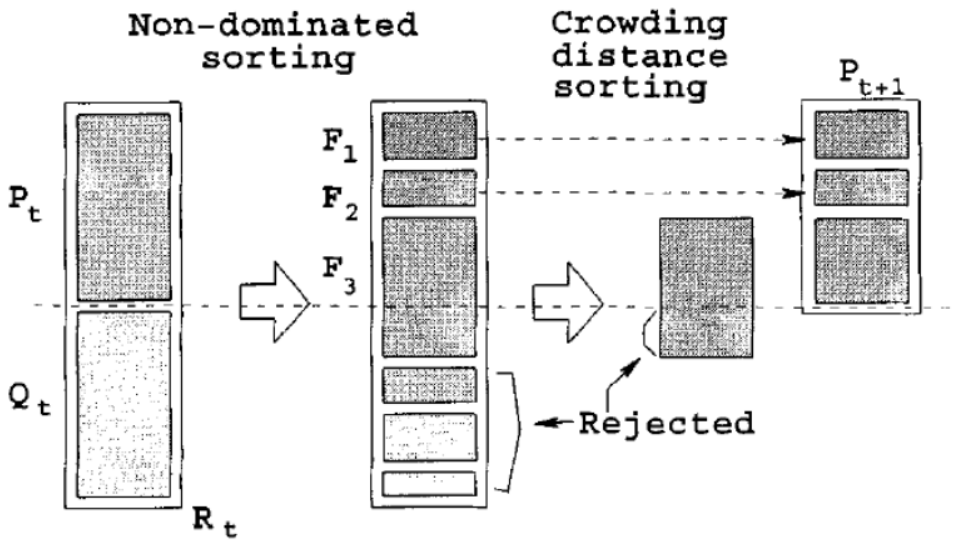


Figura 2.5: Esquema algoritmo NSGA-II [22]

2.4. Antecedentes de Estudio

En esta sección se presentan las investigaciones utilizadas en el desarrollo del trabajo de título. Aplicadas tanto a la industria en general como específicamente al área de la industria energética solar.

2.4.1. Uso de Machine Learning para optimización

Actualmente existe un gran interés en integrar Machine Learning con métodos de optimización según el fin que corresponda y se quiera investigar. Esta integración es amplia, pudiendo abarcar diversos problemas, pero en particular para este trabajo se centrará en el uso de modelos de Machine Learning para la optimización.

En la investigación de Weichert, D et al. [23] se realiza un estudio de el uso de Machine Learning con optimización para procesos de producción, donde se menciona que la metodología planteada también puede ser utilizada para ahorro de energía, tiempo, recursos y evitar desechos en lo posible. En este paper se identifican dos posibles enfoques de uso de Machine Learning, sin cambios de parámetros y con cambios de parámetros. Donde el segundo caso se basaría en generar la base de datos, luego modelar el problema con Machine Learning, es decir, entrenar modelos de Machine Learning capaces de predecir el funcionamiento físico del problema a plantear, para terminar con la optimización de los parámetros usando el modelo creado.

También en la investigación de Lee, J. et al. [24], se utiliza una metodología similar a la mencionada anteriormente para la optimización de producción de hidrógeno que se puede observar en la figura 2.6. Se utilizan algoritmos de Machine Learning capaces de predecir el

funcionamiento de una planta de hidrógeno, para realizar una optimización según los outputs del modelo. En este caso en particular la metodología utiliza tuner de hiperparámetros, lo que significa buscar la mejor combinación de hiperparámetros para lograr obtener un modelo capaz de predecir con bajo error las variables dependientes, para terminar con la optimización del problema basado en los outputs obtenidos.

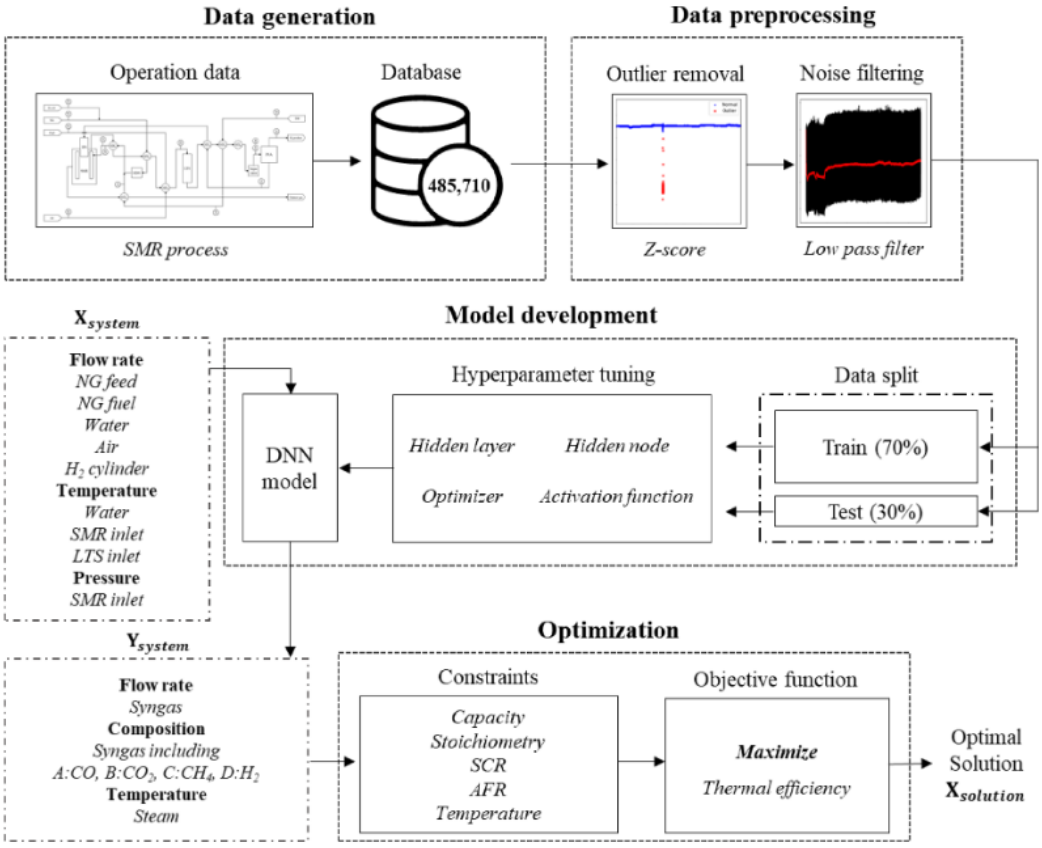


Figura 2.6: Esquema metodología investigación Lee, J et al. [24]

Esta metodología también se ocupa en la investigación de Magnier, L. et al [25]. Donde su objetivo corresponde a optimizar el diseño de edificios, en particular confort térmico y consumo de energía de una casa residencial. Para esto en primer lugar obtiene una base de datos simulados con TRNSYS, mismo programa que se puede utilizar para simular plantas solares, para luego entrenar una Red Neuronal y optimizar mediante Algoritmos Genéticos.

La misma metodología es aplicable a diversos casos de estudio. Como se mencionó en [23], esta metodología también puede ser utilizada al área energética. Es importante mencionar también que en [23] se mencionan pasos importantes previos al uso de Machine Learning, como la identificación de los datos a trabajar; es decir si corresponden a datos cualitativos o cuantitativos, series de tiempo, etc. Además, para estos trabajos es importante identificar el origen de los datos, si son medidos o simulados y también identificar si se necesitan limpiezas o preprocesamientos. Lo que significa que para este tipo de trabajos no sólo es importante la buena aplicación de los algoritmos, si no que también hay que tener sumo cuidado con la

generación y el trabajo previo de los datos a utilizar.

2.4.2. Machine Learning aplicado a energía solar

Como se mencionó en la sección 2.2, es necesario optimizar las variables de la planta solar para en primera instancia minimizar los costos y lograr que el proyecto sea rentable. La implementación de modelos de Machine Learning para la optimización puede ser aplicado a diversas áreas, y una en particular corresponde a la producción de energía. En este caso para la energía solar térmica.

En la investigación de Fadlallah, S. et al. [26] desarrollan un algoritmos de Redes Neuronales para la predicción y Particle Swarm Optimization (PSO) como algoritmo de optimización para analizar y optimizar el uso de material combinado "Sandwich" para la estructura de heliostatos. En esta investigación se diseña la Red Neuronal a utilizar creando diversas arquitecturas en un loop hasta obtener un error MSE definido lo suficientemente bajo. Para luego utilizarlo en el algoritmo PSO, donde la población inicial serían los inputs de la Red Neuronal obteniendo como output el fitness de cada uno de las posibles soluciones, hasta satisfacer la condición definida y obtener el óptimo global.

En el caso de la investigación de Kalogirou, S. [11], aplica la metodología de una forma más simple, utilizando un objetivo y optimizando con Algoritmos Genéticos, para optimizar el tamaño del tanque de almacenamiento de energía auxiliar y la correlación con el área del colector. Para esto en primer lugar obtiene una base de datos utilizando TRNSYS, y con esto entrena un modelo de Group Method of Data Handling (GMDH), que corresponte a un tipo de Redes Neuronales, y luego optimizar utilizando Algoritmos Genéticos.

De forma análoga, en la investigación de Dallapiccola, M. et al. [27], como también la investigación de Starke, A. et al. [28], aplican el mismo modelo de Kalogirou, S., pero con una optimización multiobjetivo y para esto utilizan el algoritmo NSGA-II. En el caso de estas investigaciones se busca optimizar plantas híbridas de colectores solares y paneles fotovoltaicos aprendiendo sus outputs con modelos de Machine Learning. La diferencia principal de estas investigaciones es que Starke, A. et al. al igual de Kalogirou, S. utiliza Redes Neuronales, mientras que el modelo ocupado por Dallapiccola, M. et al. corresponde a Árboles de Decisión. En la Figura 2.7 adaptada de [27] se puede observar un esquema general de las metodologías presentadas.

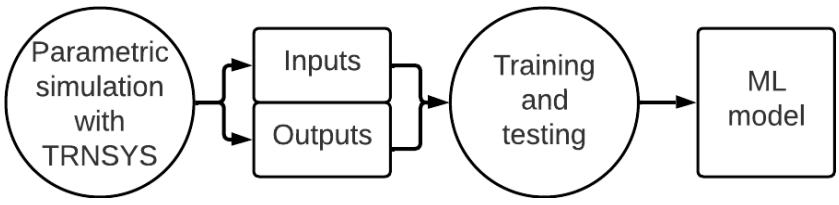


Figura 2.7: Esquema del modelo de aprendizaje y optimización [27]

Por lo tanto, se pueden usar los algoritmos mencionados anteriormente para lograr op-

timizar una planta solar térmica. Para esto en primer lugar es necesario generar una base de datos con las variables dependientes e independientes de plantas anteriormente diseñadas modeladas en el software TRNSYS 18. Luego estos se utilizan para entrenar algún modelo de Machine Learning y si estos tienen buena capacidad de predicción se utilizan algoritmos genéticos para optimizar las variables dependientes. Dependiendo de si el modelo de optimización es de un o multi objetivo, se utilizaría el algoritmo genético simple o NSGA-II respectivamente.

Capítulo 3

Metodología

A continuación, se presenta la metodología utilizada en el desarrollo del trabajo, con sus respectivas etapas y los pasos realizados en cada una de ellas.

3.1. Generación y limpieza de datos

Para entrenar los algoritmos de Machine Learning en primer lugar es necesario contar con un set datos o dataset. En particular para este trabajo se utilizaron dos, los cuales fueron simulados para la planta solar térmica que se puede observar en la figura 3.1 con el programa TRNSYS 18 por Centro Fraunhofer Chile Research, para posteriormente ser entregados para la realización del trabajo.

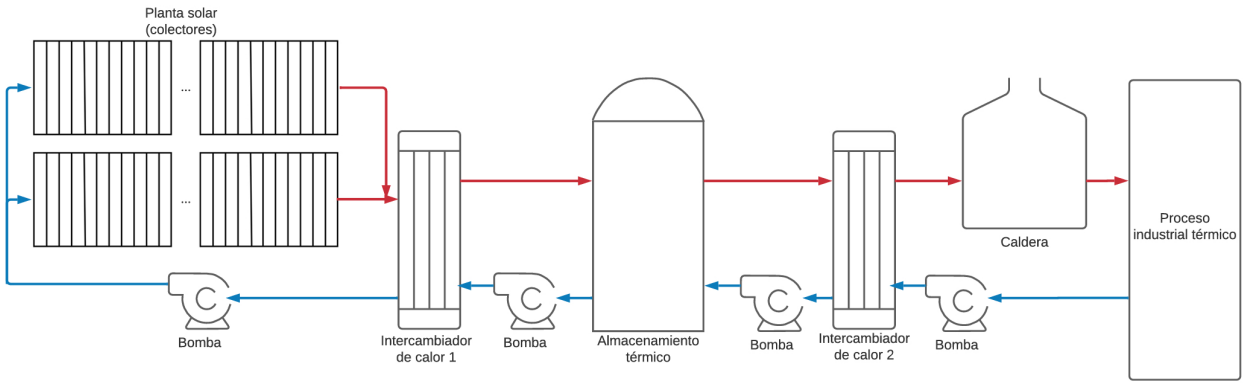


Figura 3.1: Diagrama de flujo de planta solar térmica

Ambos datasets cuentan con cinco variables independientes y dependientes, y también un comprobador si el conjunto de datos es consistente según la energía generada. Por esto, el primer paso a realizar corresponde a aislar los datos inconsistentes e identificar sólo los útiles para el problema. En la tabla 3.1 se pueden observar los datos totales y los útiles luego de la limpieza.

Tabla 3.1: Cantidad de datos para cada dataset

Dataset	Datos totales	Datos útiles	Porcentaje [%]
Primero	7982	7761	97.23
Segundo	35570	27559	77.48

Luego de obtener sólo los datos útiles, es necesario identificar cuáles de estos corresponden a las variables independientes, los inputs del problema (***"X"***) y variables dependientes, los outputs (***"Y"***), realizando la separación correspondiente. Ambos datasets tienen las mismas variables, pero los inputs pertenecen a diferentes conjuntos. Teniendo que el primer dataset es homogéneo, es decir, con datos uniformes equiespaciados, mientras que el segundo heterogéneo, donde los datos tienen más variación entre sí. Esto se puede visualizar de mejor forma a continuación donde se identifican los conjuntos para cada dato tanto para el primer como segundo dataset enumerados.

- ***Área del campo solar*** (Area_f) [m^2]: Área que ocupan los colectores solares.
  1. $Area\_f$ = {4000, 5000, 6000, 7000, 8000, 9000, 10000}
  2. $Area\_f$ = A = [4000, 60000]
- ***Volumen del estanque*** (TES_V) [m^3]: Volumen del estanque de almacenamiento, calculado con el ***volumen específico*** (Spec_Vol) [m^3/m^2] según la ecuación 3.1.

$$TES\_V = Area\_f \cdot Spec\_Vol \ \ [m^3] \tag{3.1}$$

  1. $Spec\_Vol \in$ {0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225}
  2. $Spec\_Vol \in$ [0.05, 0.35]
- ***Ángulo de inclinación*** (β) [°]: Ángulo con el cuál están inclinados los colectores solares de la planta.
  1. $\beta \in$ {0, 10, 20, 30, 40, 50}
  2. $\beta \in$ [0, 60]
- ***Flujo másico*** (sf_m) [kg/hm^2]: Flujo másico que circula por el campo solar, es decir, por los colectores al primer intercambiador de calor.
  1. sf_m $\in$ {40, 50, 60, 70, 80}
  2. sf_m $\in$ [40, 100]
- ***Efectividad intercambiadores de calor*** (ep_hx) [-]: Corresponde a la efectividad de los intercambiadores de calor, donde se asume que para ambos es igual.
  1. ep_m $\in$ {0.5, 0.6, 0.7, 0.8, 0.9}
  2. ep_m $\in$ [0.5, 0.9]

Las variables dependientes corresponden a los outputs. Estos corresponden a diversas energías por año asociadas y generadas por la planta solar, por lo que ambos datasets cuentan con los mismos outputs.

- ***Irradiación que llega al campo solar*** (E_inc) [kWh/año]: Energía que llega por año a los colectores del campo solar.
- ***Energía transferida al fluido*** (Eu) [kWh/año]: Energía que el campo solar efectivamente transfiere al flujo másico circulante por año.
- ***Energía del intercambiador de calor 1*** (EHX1) [kWh/año]: Energía transferida del lado caliente al frío en el intercambiador de calor 1 por año.
- ***Energía del intercambiador de calor 2*** (EHX2) [kWh/año]: Energía transferida del lado caliente al frío en el intercambiador de calor 2 por año.
- ***Energía de la caldera*** (E_heater) [kWh/año]: Energía auxiliar generada por la caldera de la planta solar por año.

3.2. Preprocesamiento y segmentación de los datos

Sabiendo que los preprocesamientos son una etapa esencial en el uso de algoritmos de Machine Learning, es necesario aplicarlos para el buen funcionamiento de los modelos. Se aplicaron cinco tipos y también se estudió el caso base con los datos originales. Esto se realizó para comparar y analizar cual caso es mejor para el problema estudiado. Es por esto que se aplicaron los preprocesamientos MinMax Scaler, MaxAbs Scaler y Standard Scaler, mostrados en las ecuaciones 2.2 a 2.4, y se aplicó Análisis de componentes principales (PCA) a 2 y 3 dimensiones. Para aplicar todos estos tipo de preprocesamientos existen funciones implementadas en ScikitLearn [29].

Luego, es necesario dividir el set de datos para realizar los entrenamientos y posteriormente testear los modelos de Machine Learning, lo que se logra mediante la función *"train_test_split"* de Scikit-Learn. Lo que hace en particular esta función es dividir el set de datos en dos, según un porcentaje dado.

En este trabajo se realizaron dos modelos para comparación, Redes Neuronales Artificiales y Árboles de Desición. Para el caso de Redes Neuronales Artificiales, como se mencionó en la sección 2.3.2, es necesario obtener tres conjuntos del dataset, el conjunto de entrenamiento (*"Y_train"*), de validación (*"Y_val"*) y el de prueba (*"Y_test"*). Esto se realiza utilizando dos veces la función mencionada, primero se segmenta *"X"* e *"Y"* en el conjunto de entrenamiento y testeo, obteniendo el conjunto de test como un 10 % de los datos totales, y luego el conjunto de entrenamiento se vuelve a segmentar, para obtener de una parte el conjunto de validación, como el 20 % de los datos restantes. La cantidad de datos para cada conjuntos se puede observar en la Tabla 3.2 para cada uno de los datasets.

Tabla 3.2: Cantidad de Datos para cada set en Redes Neuronales

Dataset	Datos útiles	X_train	X_test	X_val
Primero	7761	5587	777	1397
Segundo	27559	19842	2756	4961

Mientras que para el caso de Árboles de Decisión no se utiliza el conjunto de validación, por lo que sólo se debe usar la función una vez, separando el conjunto de entrenamiento y prueba. Como se requieren menos conjuntos se utilizó un porcentaje más grande para obtener el conjunto de prueba. El porcentaje utilizado corresponde al 30 % de los datos totales. La cantidad de datos para cada conjuto utilizado en Árboles de Decisión se pueden observar en la Tabla 3.3 para cada dataset.

Tabla 3.3: Cantidad de Datos para cada set en Árboles de Decisión

Dataset	Datos útiles	X_train	X_test
Primero	7761	5432	2329
Segundo	27559	19291	8268

Es importante mencionar que estas segmentaciones de los datos se realizan después de aplicar el preprocesamiento a $\mathbf{X}$, esto para que todos los datos queden normalizados o reducidos bajo la misma función y no exista ningún tipo de inconsistencias.

3.3. Modelos y tuner de hiperparámetros

El siguiente paso corresponde a armar los modelos de Machine Learning, o sea los modelos de Redes Neuronales y Árboles de Decisión. Para esto se utilizan tuners de hiperparámetros. Los tuners de hiperparámetros se basan en encontrar que modelo se comporta mejor para el problema estudiado, buscando combinaciones de hiperparámetros según conjuntos dados. Es decir, para cada hiperparámetro que se quiera buscar, es necesario asociarle un conjunto de búsqueda.

3.3.1. Tuner de Redes Neuronales

Para comenzar a desarrollar la arquitectura del modelo de Redes Neuronales, en primer lugar es necesario definir una función que arme el modelo con los hiperparámetros a seleccionar. En el caso de este trabajo los hiperparámetros de búsqueda corresponden a la cantidad de capas ocultas, la cantidad de nodos y la función de activación de cada una de ellas, como también el learning rate. En la Tabla 3.4 se pueden observar los conjuntos de búsqueda para cada uno de estos.

Tabla 3.4: Conjunto de busqueda para Tuner Redes Neuronales

Hiperparámetro	Conjunto de búsqueda
Nº Capas	1, 2, 3
Nº Nodos	4, 16, 32, 64, 128, 256, 512
Función activación	relu, linear, selu
Learning rate	1e-4, 1e-3, 1e-2, 1e-1, 1e-6

Con la librería Keras [30] de Tensorflow, se inicializa el modelo secuencial y se agregan las capas correspondientes, la capa de entrada con la cantidad respectiva de inputs según el preprocesamiento que se ocupe, las capas ocultas y capa de salida con 5 nodos para los outputs. En la misma función definida también se compila el modelo, con optimizador Adam de la librería Keras y con pérdida Mean Absolute Percentage Error (MAPE) 3.2.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \widehat{y_i}}{y_i} \right| [\,\%] \tag{3.2}$$

Donde y_i corresponde a el valor simulado del conjunto de entrenamiento o validación, mientras que $\widehat{y_i}$ corresponde al valor que va prediciendo la Red Neuronal.

Al tener la función completamente definida, se inicializa el tuner con sus parámetros correspondientes, esto con la librería Keras-Tuner [31] que es un anexo para Keras. El tuner utilizado corresponde a RandomSearch con 1000 trials, es decir, se entrenan 1000 combinaciones de hiperparámetros al azar con el objetivo de minimizar la pérdida del conjunto de validación (*"val_loss"*) y con 3 re-entrenamientos de validación cruzada. Por último se inicializa el tuner, donde se identifican los conjuntos de entrenamiento y validación que fueron segmentados previamente, batch size de 30, por 100 épocas. Esto con callback early stopping, con monitor la pérdida del conjunto de validación, paciencia de 10 épocas y baseline de 70 % para disminuir el tiempo de ejecución del tuner en lo posible.

En la Figura 3.2 se puede observar el pseudo-código para la definición de la función que arma el modelo de Redes Neuronales y la inicialización del tuner. Donde h_par corresponde a los hiperparámetros a buscar, h_par.Layer, h_par.Nodos, h_par.Activacion y h_par.LearningRate los conjuntos de búsqueda para cada uno de estos.

```
def modelo_build(h_par):

    model = modelo secuencial

    model.add( Capa de entrada )

    for capas in range( h_par.Layers ):
        Capa Dense( units=h_par.Nodos,
                    activation= hp_par.activacion )

    model.add( Capa de salida )

    opt = keras.optimizador( h_par.LearningRate )

    model.compile( opt, loss, metrics )

    return model

tuner = RandomSearch( modelo_build,
                      objetivo, max pruebas,
                      executions_per_trial )

tuner.search( train data, batch, epochs,
              validation data, callback )
```

Figura 3.2: Pseudo-código Tuner de Hiperparámetros Redes Neuronales

3.3.2. Tuner de Árboles de Decisión

Para el caso de Árboles de decisión primero se definen los hiperparámetros con sus respectivos conjuntos de búsqueda en un diccionario. Los utilizados fueron el criterio (criterion), el número mínimo de samples para particionar (min_samples_split) y el splitter. Se utilizaron sólo estos, porque corresponden a hiperparámetros de decisión, mientras que otros son para restringir el crecimiento del árbol. En la tabla 3.5 se pueden observar los conjuntos de búsqueda de estos.

Tabla 3.5: Conjunto de busqueda para Tuner Árboles de Decisión

Hiperparámetro	Conjunto de búsqueda
Criterion	squared_error, friedman_mse, absolute_error, poisson
Min samples split	2, 3, 4, 5, 6, 7, 8, 9, 10
Splitter	best, random

En primer lugar, se inicializa el modelo Árbol de Decisión para regresión. Luego, se define el tuner de hiperparámetros que corresponde a la función GridSearch de Scikit-Learn, el cual realiza los entrenamientos de todos los modelos posibles según la combinación de hiperparámetros dentro de los conjuntos de búsqueda. Este se define con el modelo de Árbol de regresión inicializado previamente, el diccionario de hiperparámetros, scoring el cual al igual que Redes Neuronales se define como el Mean Absolute Percentage Error definido en la ecuación 3.2. Además el tuner se definió con 5 validaciones cruzadas. Por último, se inicia el tuner donde con este conjunto de búsqueda se tienen 72 modelos, y así por la validación cruzada se entrenan 360 modelos en total. En la Figura 3.3 se puede observar el pseudocódigo

de lo detallado donde los datos de entrenamiento se refieren a *"X_train"*.

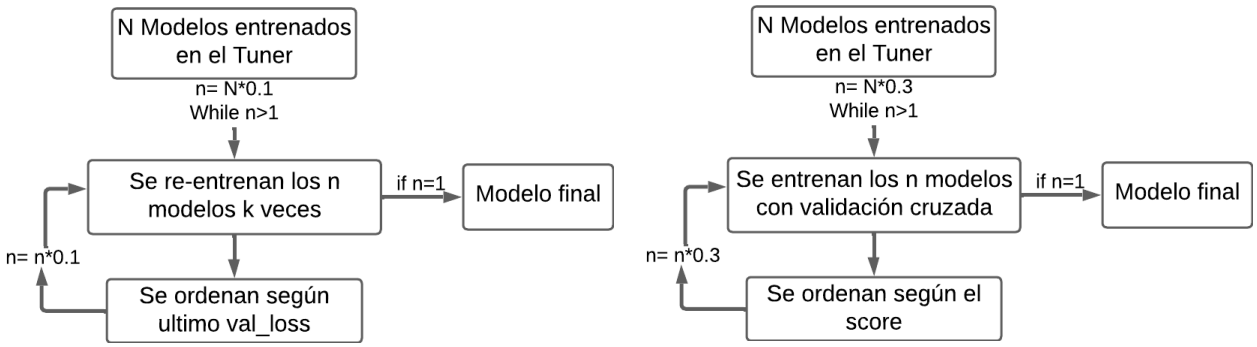
```
reg_model = DecisionTreeRegressor()

parameters = { 'criterion',
               'min_samples_split',
               'splitter' }

tuner = GridSearchCV( reg_model,
                      parameters,
                      scoring,
                      cv )

tuner.fit( datos de entrenamiento )
```

Figura 3.3: Pseudo-código Tuner de Hiperparámetros Árboles de Decisión

3.4. Selección del modelo final

Para obtener el modelo final con el que se realizará la optimización de la planta solar térmica, se realiza un torneo de modelos inspirado en [32]. Este torneo corresponde a ir re-entrenando los mejores modelos obtenidos e ir comparando según menor pérdida de validación, hasta quedar con el modelo final.

Este torneo se realiza mediante un ciclo while, donde se toma el 10 % mejor según la pérdida de validación en el caso de Redes Neuronales y el 30 % según el score de validación cruzada en el caso de Árboles de Decisión, siempre aproximando hacia arriba. Estos modelos se re-entrenan y se vuelve a repetir el paso anterior hasta quedar con un único modelo. El esquema con las diferencias entre Redes Neuronales 3.4.a y Árboles de Decisión 3.4.b se pueden observar en la figura 3.4.



(a) Diagrama para Redes Neuronales

(b) Diagrama para Árboles de Decisión

Figura 3.4: Selección de mejores modelos de Machine Learning

En el caso de redes neuronales, se re-entrenan los modelos de los hiperparámetros correspondientes utilizando EarlyStopping con paciencia de 20 épocas, y un delta mínimo variable según la cantidad de modelos que se están entrenando, además para evitar que los modelos dependan de los pesos iniciales en cada torneo se re-entrenan los modelos k veces dependiendo de la cantidad de modelos. Es decir, si se tienen más de 40 modelos se tiene un delta de 0.01

con un k de 4, para más de 10 modelos un delta de 0.001 con k de 5 y por último para más de un modelo un delta de 0.0001 con k de 7. Además, cada uno de estos modelos se entrenan por 200 épocas, con batch size de 30.

Mientras que en el caso de Árboles de decisión, se calculan los scores de validación cruzada utilizando la función LearningCurve de ScikitLearn, que toma como parámetros el modelo de regresión con los hiperparámetros seleccionados en orden, el vector training sizes y la cantidad de validaciones cruzadas con scoring MAPE. La cantidad de validaciones cruzadas utilizadas es variable dependiendo de la cantidad de modelos en el torneo, donde para más de 40 modelos se usa 5, para más de 10 se utilizan 7 y para más de 1 serían 10.

3.5. Obtención y comparación de resultados

Al obtener el último modelo se obtienen los resultados pertinentes. En primer lugar, el último modelo se re-entrena 10 veces y con las últimas pérdidas de cada uno se calculan las desviaciones estándar y los promedios. Esto para poder observar las estadísticas del modelo y poder asegurar que no depende de la inicialización de pesos de la Red Neuronal

Luego, se continua el entrenamiento del modelo por 2000 épocas más y con un EarlyStopping con paciencia de 50 épocas, y un delta mínimo de 0.00001. Con el modelo ya completamente entrenado se obtiene el gráfico de pérdidas y se evalúa el modelo con el conjunto de prueba. Con estos resultados, se calcula la pérdida total, como también las pérdidas de cada uno de los outputs, para observar en que porcentaje divergen los resultados predichos por la red (***"Y_pred"***) y los del set de datos de prueba (***"Y_test"***).

Luego, se grafica *"Y_test"* v/s *"Y_pred"* con un margen de error del 5 %. Como ambos valores deberían ser lo suficientemente iguales, este gráfico debería ser la línea de la identidad, es decir, estar en un ángulo de 45° del eje x. Esto para observar de manera gráfica la divergencia entre los valores, con el respectivo margen de error y así identificar que tan bien predicen los resultados el algoritmo de Machine Learning con su respectivo preprocesamiento realizado y distinguir si los resultados son los deseables. Por último se obtienen los hiperparámetros del problema, para observar cuál fue el mejor modelo de las combinaciones en el tuner.

Cada uno de estos pasos mencionados para armar los modelos de Machine Learning, tanto Redes Neuronales como Árboles de Decisión, se realizan para cada uno de los preprocesamientos mencionados. Es decir, los códigos y modelos mencionados se aplican de la misma manera 6 veces, sólo variando la transformación de **"X"** que se utiliza mencionadas previamente.

3.6. Optimización para un objetivo

Luego de la obtención de los modelos finales, se identifica cual de estos tienen menor error y mejor capacidad de predicción. Y estos son utilizados para realizar la optimización con Algoritmos Genéticos. Se selecciona el mejor preprocesamiento obtenido para cada modelo,

es decir, para Árboles de Decisión y Redes Neuronales para el primer y segundo dataset. Para la optimización de un objetivo utilizando Algoritmos Genéticos se utilizó la librería PyGad de Python [21].

El problema de optimización quedaría definido como la minimización del LCoH, mostrado en la ecuación 2.1, de la planta solar térmica con la cual se está trabajando sujeto a que la eficiencia de la planta y fracción solar debe ser mayor al 50 %. Además se realiza un análisis de sensibilidad dependiendo del precio del combustible (costo fuel) para 30, 60 y 90 $[USD/MWh]$, como también un análisis para el costo de los colectores para 200, 250 y 300 $[USD/m^2]$. Obteniendo 9 optimizaciones para cada modelo escogido.

Es importante notar que para el uso de los modelos de Machine Learning es necesario transformar los individuos según el preprocesamiento correspondiente. Es por esto que antes de definir la función, es necesario utilizar el dataset original de datos para obtener el scaler. Esto para que al ir transformando los individuos de forma individual tengan la misma forma que los datos que se usaron en el entrenamiento de los modelos.

3.6.1. Función Fitness

La función fitness del algoritmo depende de el modelo de Machine Learning obtenido y valores constantes que corresponden a los costos, años de estudio y la tasa de descuento. Los que se muestran a continuación:

- Costo colectores (C_{col}) = 200, 250 y 300 [USD/m^2]
- Costo estanque (C_{est}) = 1000 [USD/m^3]
- Costo combustible (C_{fuel}) = 30, 60 y 90 [USD/MWh]
- Años de estudio (T) = 20 [años]
- Tasa de descuento (r) = 7 [%]

Para el caso del primer dataset el segundo gen del individuo corresponde al volúmen del estanque, es decir, directamente el input del modelo de Machine Learning, mientras que en el caso de el segundo dataset se trabaja con el volumen específico, que luego se multiplica por el área de colectores para obtener el input del volumen. Así, se define la función fitness siguiendo los siguientes pasos a continuación:

1. Se transforma el individuo según el preprocesamiento que corresponda.
2. Se identifica cada uno de los inputs, es decir, el Area_f, TES_V, beta, sf_m y ep_hx.
3. Se predicen las energías generadas por la planta utilizando el modelo de Machine Learning para el individuo.
4. Se identifica cada output, análogo al paso 2, para facilitar cálculos.

5. Se calcula la demanda energética (E_t) del proceso industrial aproximado como la suma de la energía de la caldera y del intercambiador de calor 2.

$$E_t = E\_heater + EHX2 \; [kWh/\text{año}] \tag{3.3}$$

6. Se calcula la inversión inicial (I_0) de la planta solar, esto con los costos totales de los colectores solares, del estanque y sumando la aproximación de mano de obra como el 20 % de los valores mencionados.

$$I_0 = 1.2(C_{col} \cdot Area\_f + C_{est} \cdot TES\_V) \; [USD] \tag{3.4}$$

7. Se calculan los costos de operación (C_t) como el 2 % de la inversión inicial más el costo del combustible por año.

$$C_t = 0.02 \cdot I_0 + C_{fuel} \cdot E\_heater \cdot 10^{-3} \; [USD] \tag{3.5}$$

8. Luego, utilizando ciclo for, se calculan las sumatorias correspondientes a la ecuación del LCoH, es decir, la de costos de operación y la de energías.

9. Se calcula el LCoH con los valores obtenidos de las sumatorias e inversión inicial con la ecuación 2.1.

10. Como el fitness de un individuo siempre quiere ser maximizado, y el LCoH quiere ser minimizado, se identifica la función fitness cómo:

$$Fitness = \frac{1}{1 + LCoH} \tag{3.6}$$

11. Por último, como el problema está restringido, se penaliza el fitness restando un valor N lo suficientemente grande, si el individuo no cumple con las condiciones. La eficiencia de la planta solar (EF) y fracción solar (FS) se definen de la siguiente manera:

$$EF = \frac{Eu}{E\_inc} \tag{3.7}$$

$$FS = \frac{EHX2}{E_t} \tag{3.8}$$

3.6.2. Población inicial

Se debe crear la población inicial para comenzar con el algoritmo. Esta se inicializa con N individuos, cada uno con genes al azar según corresponda dentro de sus rangos. Esto se realiza mediante un ciclo for para la cantidad de individuos. Es importante notar que todas las generaciones tienen la misma cantidad de individuos, es decir, la elección de N para la población inicial identifica también la cantidad de individuos para todas las generaciones.

Para este trabajo se utilizaron generaciones de N=500 individuos.

3.6.3. Función Mutación

Como cada uno de los inputs pertenecen a diferentes conjuntos, y el volumen depende del área de los colectores para el segundo dataset, es mejor definir la función que utilizar una predefinida por la librería.

Así, para definir la función mutación se siguen los siguientes pasos:

1. En un ciclo for para cada individuo, se asocia una probabilidad de que ocurra mutación a cada uno de sus genes (inputs).
2. En otro ciclo for para cada gen, si la probabilidad de cada uno es menor o igual a la probabilidad por definir en la inicialización del algoritmo, se realiza un cambio al azar del gen en cuestión, dentro de sus rangos correspondientes.
3. Lo anterior se realiza para cada uno de los genes con sus respectivas probabilidades asociadas.

3.6.4. Parámetros e inicialización

Para inicializar el algoritmo es necesario definir una serie de parámetros restantes. Estos corresponden a el número de generaciones de estudio, la cantidad de padres seleccionados, el tipo de selección de padres, la cantidad de padres que pasan a la siguiente generación (Elitismo), la probabilidad de crossover y mutación, y por último, el tipo de crossover. En este caso, se utiliza una función predefinida de PyGad para esto, la que corresponde a la función "single point". Esta divide dos individuos padre en un punto, y genera dos individos juntando los genes.

Los valores de los parámetros mencionados se pueden observar a continuación:

- Número de generaciones = 500
- Número de padres = 150
- Selección de padres = "Ruleta Rusa"
- Elitismo = 25
- Probabilidad de crossover = 80 %
- Probabilidad de mutación = 30 %
- Tipo de Crossover = "Single Point"

Los valores de las probabilidades fueron escogidos observando valores típicos utilizados en [33]. Y por último se seleccionó la cantidad de padres como el 25 % de la población. Además, se utiliza un criterio análogo al early stopping, para detener el algoritmo si no hay mejoras de fitness en un total de generación dadas. En este caso se definen 200 generaciones.

Luego de tener todos los parámetros definidos, y sumando las funciones fitness y mutación como también la población inicial, se inicializa el algoritmo, con la función implementada llamada pygad.GA y se corre mediante la función run.

Por último se obtienen los resultados, con los respectivos gráficos de la evolución del fitness para las generaciones, los cambios de los genes y la composición de la planta con sus respectivas energías predichas por el modelo utilizado. Todo lo mencionado anteriormente, se puede observar en el diagrama de la metodología descrita de la Figura 3.5.

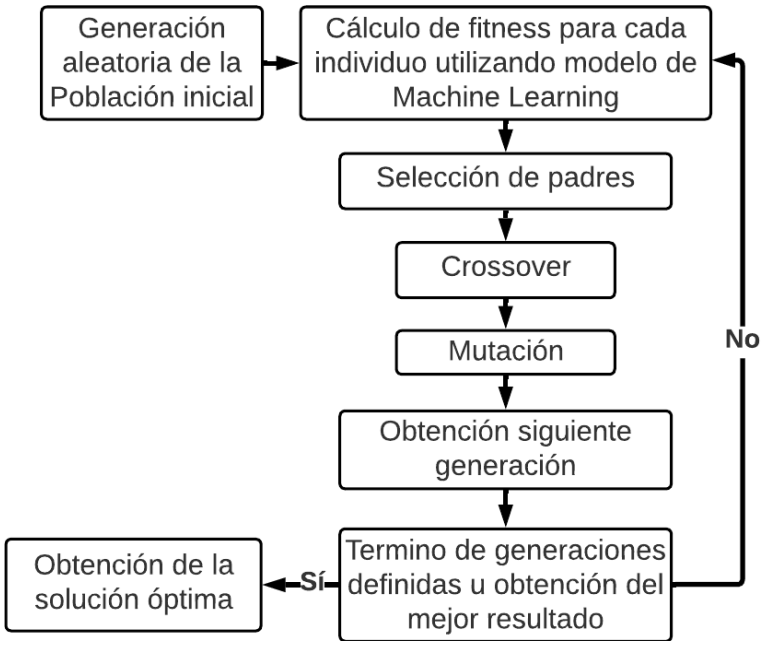


Figura 3.5: Diagrama metodología Algoritmo Genético de un objetivo

3.7. Optimización multi-objetivo con NSGA-II

Los mismos modelos de Machine Learning utilizados para la optimización de un objetivo, se utilizan para la optimización multi-objetivo. La cual, en este caso en particular, corresponde a optimización con dos funciones objetivo. Para esto se implementa el algoritmo NSGA-II de la librería pymoo para Python [34] para el problema de minimizar el LCoH de la planta y la energía de la caldera, esto restringido a que la eficiencia y fracción solar debe ser mayor o igual a 50 %.

Para esto, al igual que el caso anterior, lo primero es importar el set de datos original y realizar el fit del preprocesamiento correspondiente. Luego se definen las funciones objetivo,

restricciones, mutación, etc. En el caso de este algoritmo, estas funciones deben ser definidas como objetos para poder aplicarlas. Y se utiliza el mismo tipo de crossover que como el caso anterior, que ya viene definido en pymoo.

Es importante mencionar nuevamente, que debido a las implementaciones del problema para el caso del primer dataset el segundo gen corresponde a el volumen del estanque, es decir, directamente el input de los modelos de Machine Learning. Mientras que para el caso del segudo dataset se trabaja con el volumen especifico, el cuál luego se multiplica por el área del campo para obtener el volumen.

3.7.1. Variables, funciones objetivo y restricción

En primer lugar se define un objeto que defina el problema a realizar, el cual en este trabajo se llamó "MyProblem". Así, lo primero corresponde a definir la cantidad de variables de los individuos, es decir la cantidad de genes que en este caso corresponde a los 5 inputs del problema. Luego la cantidad de funciones objetivo, restricciones y los límites superiores e inferiores para cada gen. Para el problema se tienen 2 funciones objetivo, 1 restricción y los límites de cada gen, es decir, sus valores menores y mayores dentro de los datasets entregados.

En el mismo objeto también se define las funciones objetivo y restricciones a la cual está asociada el problema. En este caso el problema queda definido como la minimización del LCoH, al igual que la optimización de un objetivo, pero también a la minimización de la energía de la caldera, con el objetivo sea lo menor posible. Lo que se realiza para los mejores modelos obtenidos tanto de Redes Neuronales como para Árboles de Decisión. Los pasos a seguir para esta definición son casi iguales que para el caso anterior y se utilizan los mismos valores definidos para el precio de los colectores y estanque, tasa de descuento, años de estudio. Las definiciones se diferencian principalmente en la segunda función objetivo, como se definen la restricción y que este algoritmo siempre busca minimizar las funciones.

1. Se identifican cada uno de los inputs de los modelos según el gen del individuo que corresponda.
2. Se transforman con el scaler correspondiente anteriormente ajustado. Y utilizando estos se predicen las energías utilizando el modelo de Machine Learning.
3. Se identifica cada uno de los output según sus energías correspondientes para la realización de cálculos.
4. Se calcula la inversión del campo solar y estanque, y con esto la inversión inicial con la misma fórmula anterior 3.4.
5. Nuevamente con las ecuaciones 3.5 y 3.3 se calculan los costos de operación y demanda.
6. Se calcula el LCoH, así este y la energía auxiliar de la caldera, es decir E_heater, se definen como las funciones objetivo (F) según corresponda para la minimización.
7. Se definen la restricciones (G) de eficiencia y fracción solar.

3.7.2. Función mutación

Para un mayor rendimiento del algoritmo, la función mutación en este caso corresponde a Polinomial Mutation. Esta realiza un cambio en un gen del individuo según una probabilidad Pm dentro de cada uno de sus rangos definidos, luego calcula δ_1 y δ_2 y δ_q como se muestra en la figura 3.6. Finalmente, el nuevo gen mutado correspondería a X_i como la suma del gen anterior con el δ_q por la resta de los máximos y mínimos. El algoritmo de esta mutación se puede observar en la figura 3.6 obtenida de [35]. Es importante mencionar que la probabilidad se definió como 0.3 al igual que el caso anterior, y η como 20, siendo el valor mas utilizado en la literatura [35].

i $\leftarrow$ 0 ;
repeat
  r $\leftarrow$ U[0,1] ;
  if $r \leq P_m$ **then**
    $\delta_1 \leftarrow \frac{X_i - X_i^{Lower}}{X_i^{Upper} - X_i^{Lower}}$;
    $\delta_2 \leftarrow \frac{X_i^{Upper} - X_i}{X_i^{Upper} - X_i^{Lower}}$;
    r $\leftarrow$ U[0,1] ;
    $$\delta_q \leftarrow \begin{cases} [(2r) + (1-2r)* & \\ (1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1 & \text{if } r \leq 0.5 \\ 1 - [2(1-r) + 2.(r-0.5)* & \\ (1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} & \text{otherwise} \end{cases}$$
    $X_i \leftarrow X_i + \delta_q.(X_i^{Upper} - X_i^{Lower})$
  end
until $i++ == n$;

Figura 3.6: Algoritmo función Polinomial Mutation [35]

3.7.3. Inicialización del problema

Al tener las funciones necesarias, se debe definir el algoritmo. Esto se realiza mediante la función NSGA2 de pymoo con los parámetros necesarios. Los que corresponden al tamaño de la población, y las funciones previamente definidas. En este caso se utilizan 700 individuos por población.

Luego, se inicializa el problema con la función "minimize" de pymoo, la cual recibe el problema definido, el algoritmo NSGAII y la cantidad de generaciones de estudio, en este caso por 100 generaciones. Y por último, luego de que el problema termine de correr, se identifican los individuos como los valores obtenidos de las funciones correspondientes al frente de Pareto óptimo.

3.7.4. Identificación mejor configuración

Para identificar cual corresponde al mejor individuo dentro de la curva de Pareto se utiliza la regla del 80/20. Esta regla se basa en el principio de Pareto, donde se dice que el 20 % de los esfuerzos producen el 80 % de los resultados. En el caso de problema resuelto, se busca dar prioridad al LCoH, es decir, dar prioridad a obtener el LCoH más bajo posible. Es por esto que se identifica el 80 % del porcentaje acumulado de los valores de E_heater, que en teoría sería el porcentaje mas bajo que no tiene incidencia grande en el valor del LCoH.

3.8. Validación de resultados

Para finalizar, se realiza la validación de resultados correspondiente. Para esto se guardan las configuraciones encontradas en un archivo csv para modelar sus energías producidas mediante el programa TRNSYS-18 y compararlas con las energías predichas por los modelos de Machine Learning respectivos, como también la comparación del LCoH con estas energías modeladas.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos siguiendo la metodología planteada. Es decir, en primer lugar la selección de modelos para Redes Neuronales (ANN) y Árboles de Decisión (DT) con los dos datasets utilizados. Luego los resultados para las optimizaciones realizadas, tanto para un objetivo como para dos objetivos y por último la validación correspondiente de las configuraciones óptimas de las plantas obtenidas.

En el caso de los modelos de Machine Learning se presentan los datos obtenidos para el análisis estádístico de los 10 re-entrenamientos para cada preprocesamiento. Luego los resultados correspondientes al testear los últimos modelos obtenidos, es decir el error de evaluación y los gráficos de resultados finales para el preprocesamiento con mejor desempeño.

En el caso de la optimización de un objetivo se presenta el mejor fitness y LCoH obtenidos en todas las generaciones estudiadas y la mejor configuración obtenida. Para el caso multiobjetivo se presentan la curva de Pareto seguido por la implementación de la regla de Pareto 80/20 y nuevamente, la mejor configuración obtenida en este caso y por último se tiene el análisis de sensibilidad correspondiente para la regla utilizada. Terminando por la validación de resultados correspondiente del cálculo de LCoH con las energías predichas y nuevas modelaciones.

4.1. Selección de Modelos para primer dataset

Para el primer dataset, se presentan los resultados obtenidos para los dos modelos de Machine Learning utilizados, es decir, tanto para Redes Neuronales y Árboles de Decisión.

4.1.1. Redes Neuronales con primer dataset

En primer lugar, como se mencionó en la tabla 4.1 se presentan los resultados estadísticos obtenidos para los 10 re-entrenamientos del ultimo modelo obtenido en el tuner y torneo de hiperparámetros. Así, se presentan el promedio y desviación estándar de los últimas pérdidas obtenidas en cada re-entrenamiento, tanto para el conjunto de entrenamiento y de validación.

Tabla 4.1: Datos estadísticos Redes Neuronales con primer dataset

Estadística	SS	MMS	MAS	PCA3	PCA2	Nada
Mean val_loss [%]	0.16	4.44	8.46	3.60	5.31	29.69
Mean loss [%]	0.19	4.53	8.46	3.27	5.52	29.35
SD val_loss [%]	0.03	7.93	9.75	0.51	0.31	35.73
SD loss [%]	0.03	7.98	9.76	0.11	0.10	35.92

Posteriormente se continúa el entrenamiento del último modelo por 2000 épocas más con earlystopping, es decir, hasta que el modelo esté completamente entrenado y se evalúa el modelo con el conjunto de testeo. En la tabla 4.2 a continuación, se presentan los errores de evaluación global para el modelo, es decir el promedio para cada output, y los errores específicos por cada uno de estos.

Tabla 4.2: Errores de testeo Redes Neuronales con primer dataset

Error	SS	MMS	MAS	PCA3	PCA2	Nada
Global [%]	0.12	0.17	0.09	2.76	5.05	4.91
E_inc [%]	0.09	0.12	0.06	0.81	2.74	3.06
Eu [%]	0.12	0.18	0.09	3.22	5.52	5.22
EHX1 [%]	0.12	0.20	0.08	3.22	5.52	5.18
EHX2 [%]	0.13	0.18	0.09	3.25	5.57	5.27
E_heater [%]	0.14	0.18	0.10	3.38	5.91	5.80

Considerando las tablas anteriores, 4.1 y 4.2, se identifica el preprocesamiento StandardScaler como el modelo de mejor rendimiento. El análisis de selección de este modelo se explicará en la sección de Análisis de resultados, pero se utilizó el error global y también los datos estadístico para la decisión.

Así, a continuación se presentan los gráficos de resultados para el preprocesamiento StandardScaler. En primer lugar en la figura 4.1 se puede observar el gráfico de pérdidas, es decir la pérdida (error) porcentual versus las épocas por las cuales se entrenó la Red Neuronal, para el modelo final completamente entrenado.

En la figura 4.2 se presentan los resultados obtenidos finales de la Red, donde se muestra el gráfico de valores de outputs de testeo versus los predichos por ésta. Como estos valores en lo ideal deberían ser iguales, se observa el buen rendimiento de la Red si el gráfico se ajusta a la linea de la identidad (identity).

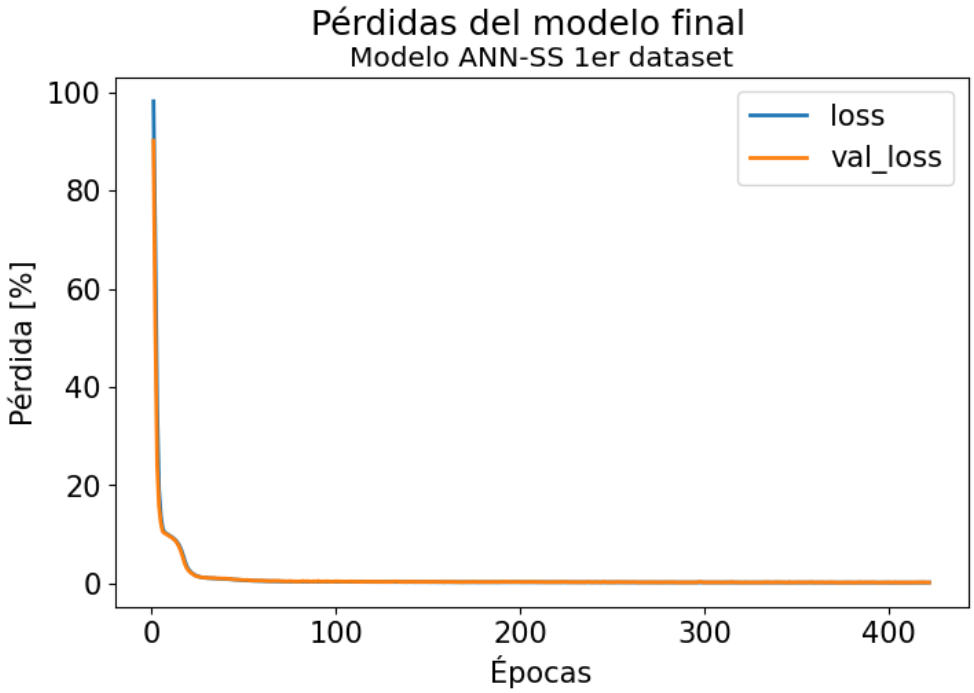


Figura 4.1: Gráfico de pérdidas modelo ANN-SS con primer dataset

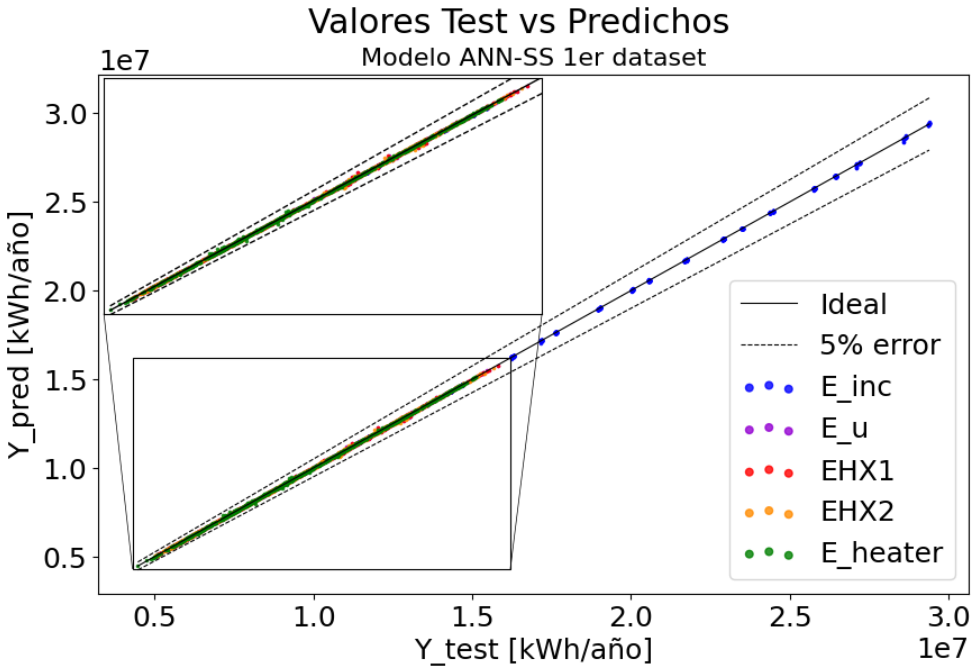
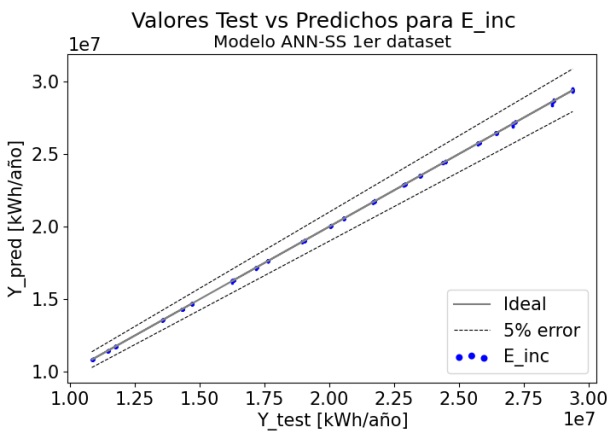
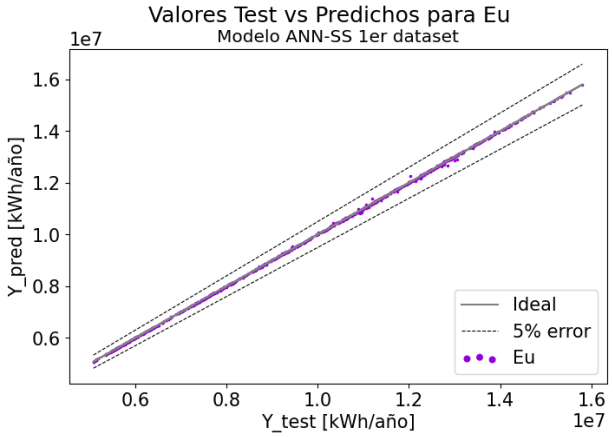


Figura 4.2: Valores test vs predichos modelo ANN-SS con primer dataset

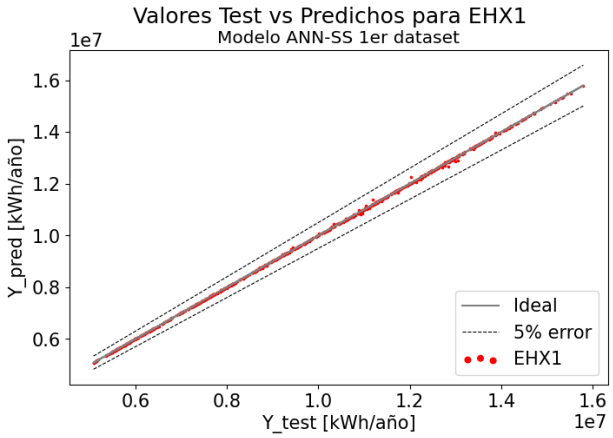
También, para una mayor comprención de los resultados, en la figura 4.3 se pueden observar los gráficos de valores de testeo versus los predichos para cada uno de los outputs en específico, es decir, cada una de las energías de la planta solar. Esto desde 4.3.a a 4.3.e.



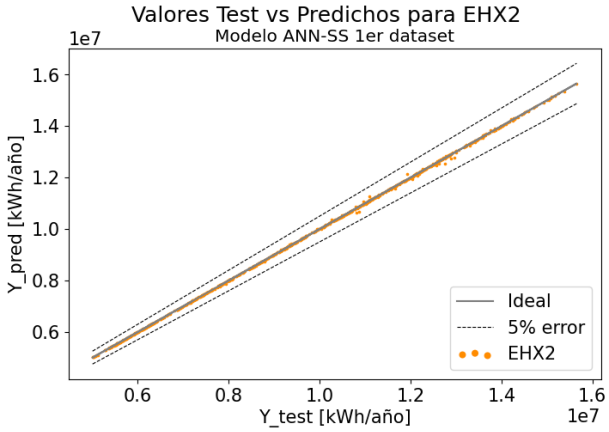
(a) Resultados E_inc



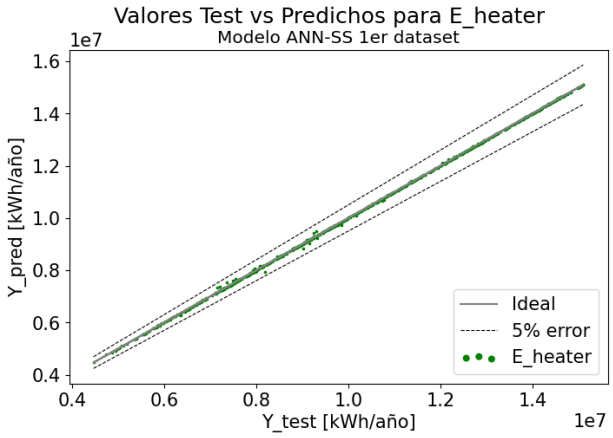
(b) Resultados Eu



(c) Resultados EHX1



(d) Resultados EHX2



(e) Resultados E_heater

Figura 4.3: Valores test vs predichos por cada output modelo ANN-SS con primer dataset

Por último para los resultados de Redes Neuronales con el primer dataset, en la tabla 4.3 se pueden observar los hiperparámetros finalmente obtenidos en la selección del modelo.

Tabla 4.3: Hiperparámetros para modelo ANN-SS con primer dataset

Ctd capas ocultas	2
Nodos capa 1	512
Activación capa 1	“selu”
Nodos capa 2	128
Activación capa 2	“relu”
Learning rate	0.01

4.1.2. Árboles de Decisión con primer dataset

Para Árboles de decición con el primer dataset en la tabla 4.4 se presentan los datos estadísticos obtenidos de los últimos scores para las 20 validaciones cruzadas realizadas.

Tabla 4.4: Datos estadísticos Árboles de Decisión con primer dataset

Estadística	SS	MMS	MAS	PCA3	PCA2	Nada
Mean Test Score [%]	0.19	0.20	0.19	2.50	3.17	0.20
Mean Train Score [%]	0	0	0	0	1.85	0
SD Test Score [%]	0.026	0.024	0.020	0.10	0.087	0.017
SD Train Score [%]	0	0	0	0	0.006	0

En la tabla 4.5 a continuación, luego de evaluar el último Árbol de Decisión, se muestran los errores obtenidos. Tanto el error global, como para cada uno de los outputs del modelo.

Tabla 4.5: Errores de testeo Árboles de Decisión con primer dataset

Error	SS	MMS	MAS	PCA3	PCA2	Nada
Global [%]	0.17	0.18	0.19	2.43	3.13	0.19
E_inc [%]	0.010	0.013	0.008	0.001	0.26	0.009
Eu [%]	0.23	0.24	0.25	3.0	3.78	0.25
EHX1 [%]	0.22	0.24	0.25	3.0	3.78	0.25
EHX2 [%]	0.16	0.18	0.19	3.03	3.81	0.18
E_heater [%]	0.21	0.24	0.25	3.03	4.04	0.24

Observando estas tablas, se puede concluir que el mejor preprocesamiento corresponde a StandardScaler. En la figura 4.4 se puede observar la Learning Curve de esta, donde se grafica el train size versus el error obtenido en el entrenamiento.

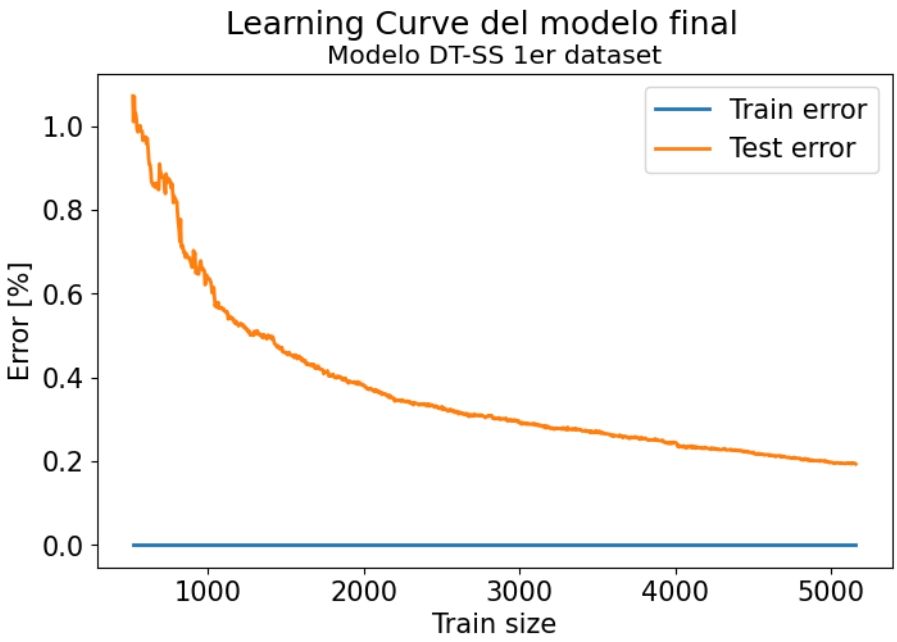


Figura 4.4: Learning Curve modelo DT-SS con primer dataset

Nuevamente, en la figura 4.5 se obsevan los resultados obtenidos para este modelo, es decir, el gráfico de los valores de testeo versus los predichos por el Árbol de Decisión con StandardScaler.

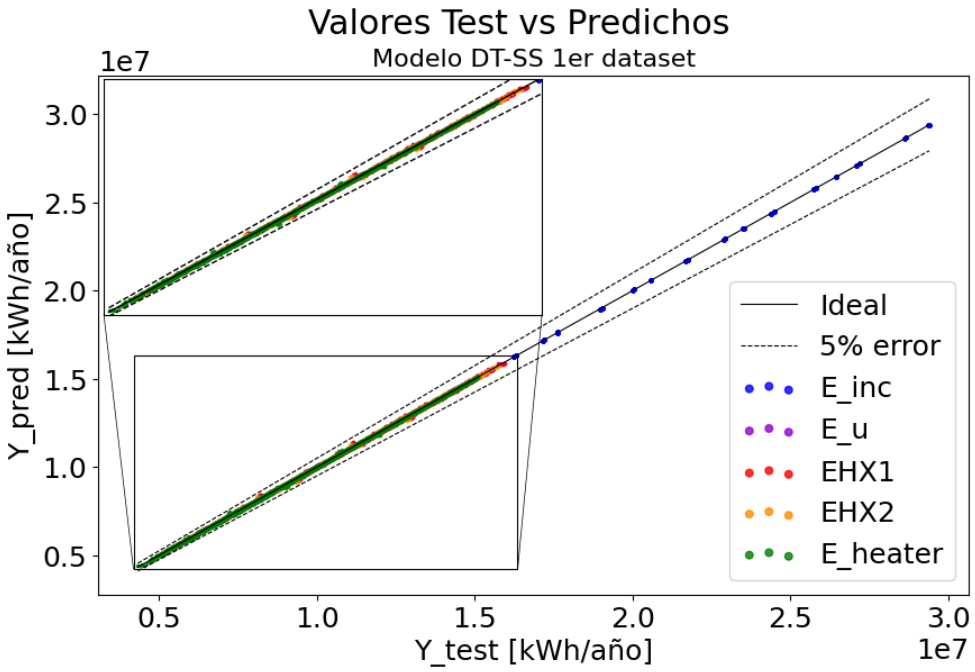
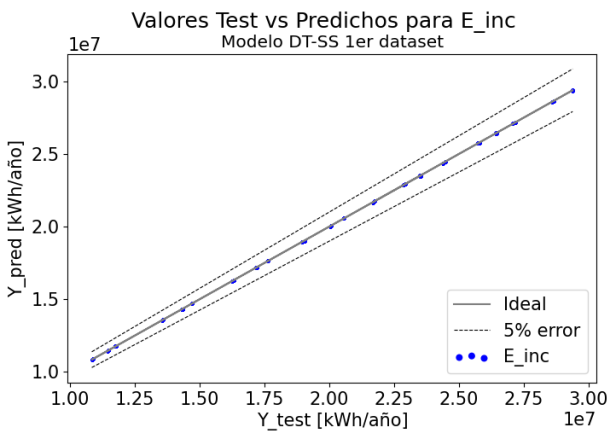
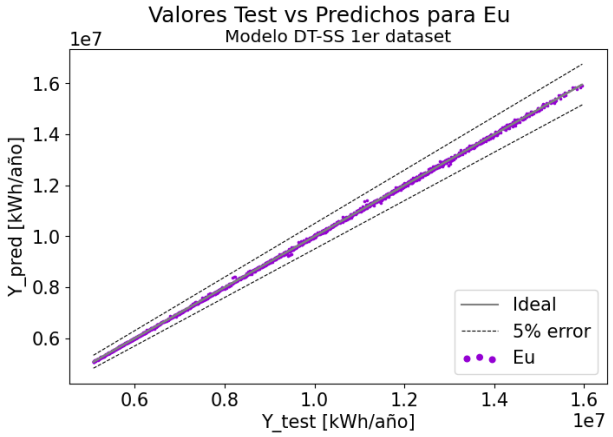


Figura 4.5: Valores test vs predichos modelo DT-SS con primer dataset

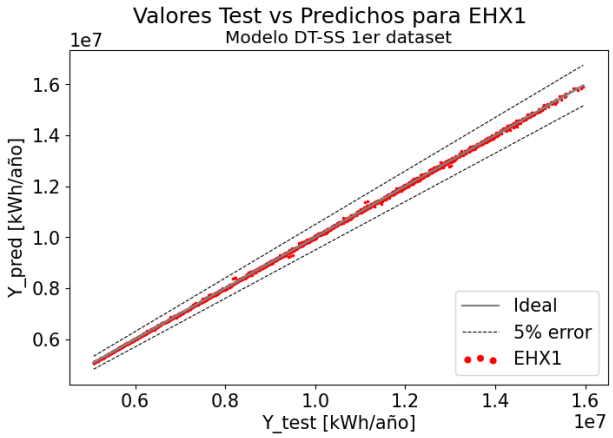
En la figura 4.6, desde 4.6.a a 4.6.e se pueden observar los graficos de valore test versus predichos para el modelo de Árbol de Decisión con el primer dataset para cada uno de los outputs en particular.



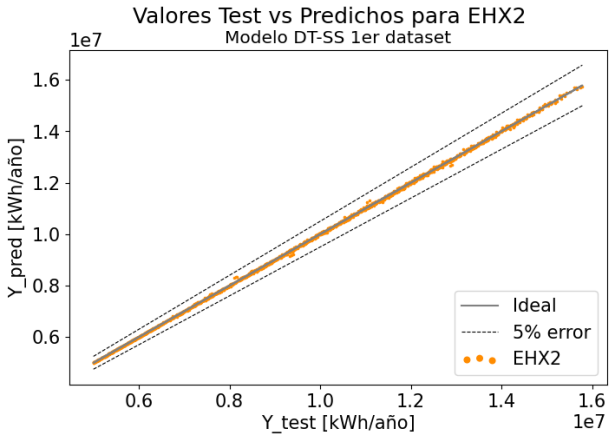
(a) Resultados E_inc



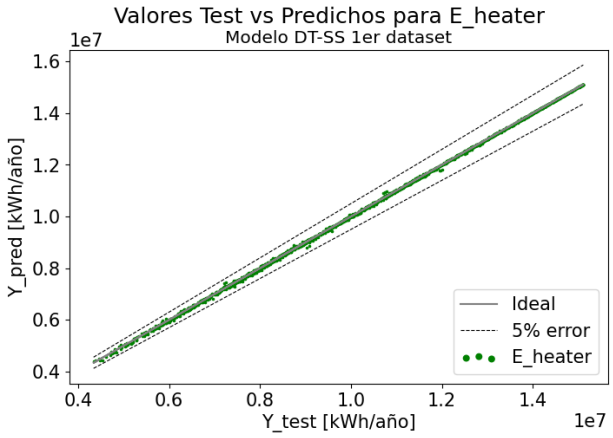
(b) Resultados Eu



(c) Resultados EHX1



(d) Resultados EHX2



(e) Resultados E_heater

Figura 4.6: Valores test vs predichos por cada output modelo DT-SS con primer dataset

Por último, se tienen los hiperparámetros obtenidos del tuner y torneo de hiperparámetros realizado para StandardScaler en la tabla 4.6 a continuación.

Tabla 4.6: Hiperparámetros para modelo DT-SS con primer dataset

Criterion	"Poisson"
min_samples_split	2
Splitter	"Best"

4.2. Selección de Modelos para segundo dataset

4.2.1. Redes Neuronales con segundo dataset

En la tabla 4.7 a continuación se pueden observar los datos estadísticos obtenidos para las Redes Neuronales para el segundo dataset, con cada uno de los preprocesamientos.

Tabla 4.7: Datos estadísticos Redes Neuronales con segundo dataset

Estadística	SS	MMS	MAS	PCA3	PCA2	Nada
Mean val_loss [%]	3.05	4.48	3.56	16.14	15.20	31.41
Mean loss [%]	3.03	5.41	3.49	5.41	14.79	31.43
SD val_loss [%]	2.39	8.02	5.81	8.02	1.80	10.02
SD loss [%]	2.17	7.81	5.84	7.81	1.54	10.04

Posteriormente, en la tabla 4.8 se tienen los errores de evaluación con el conjunto de testeo, tanto globales como para cada uno de los outputs de los modelos.

Tabla 4.8: Errores de testeo Redes Neuronales con segundo dataset

Error	SS	MMS	MAS	PCA3	PCA2	Nada
Global [%]	0.39	0.58	0.63	11.10	11.20	21.15
E_inc [%]	0.18	0.14	0.31	3.86	3.87	2.49
Eu [%]	0.13	0.14	0.18	5.38	4.92	1.12
EHX1 [%]	0.12	0.13	0.16	5.41	4.88	1.10
EHX2 [%]	0.12	0.13	0.13	5.57	5.03	1.04
E_heater [%]	1.44	2.34	2.38	35.27	37.29	100

En la figura 4.7 se puede observar el gráfico de pérdidas del entrenamiento de la Red Neuronal para el mejor modelo completamente entrenado, donde se grafican las épocas de entrenamiento versus el error de pérdida porcentual.

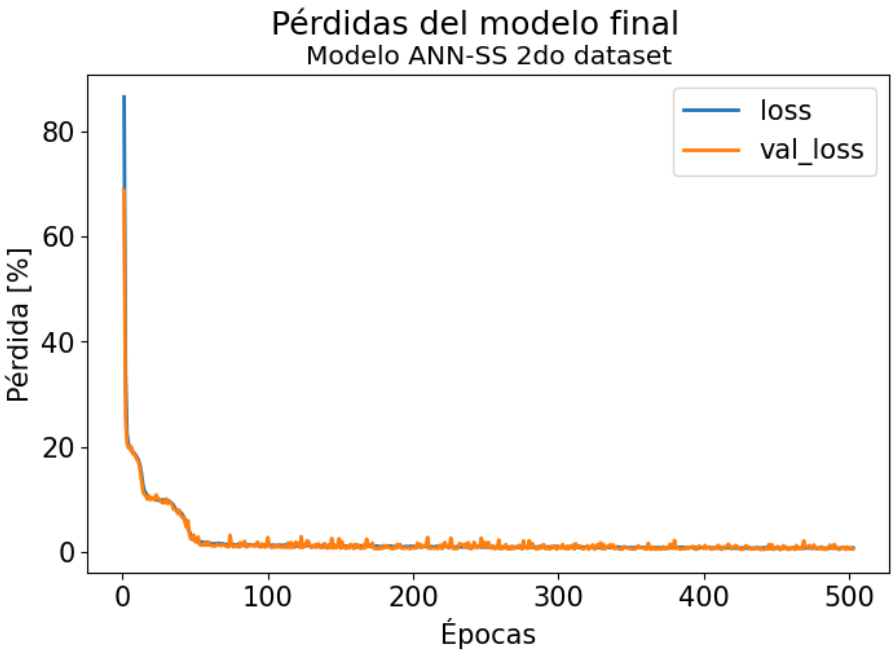


Figura 4.7: Gráfico de pérdidas modelo ANN-SS con segundo dataset

Luego, en la figura 4.8 se puede ver los resultados finales obtenidos del modelo con StandardScaler para el segundo dataset. Es decir, el gráfico de valores de testeo y predichos por la Red Neuronal para todos los outputs de ésta.

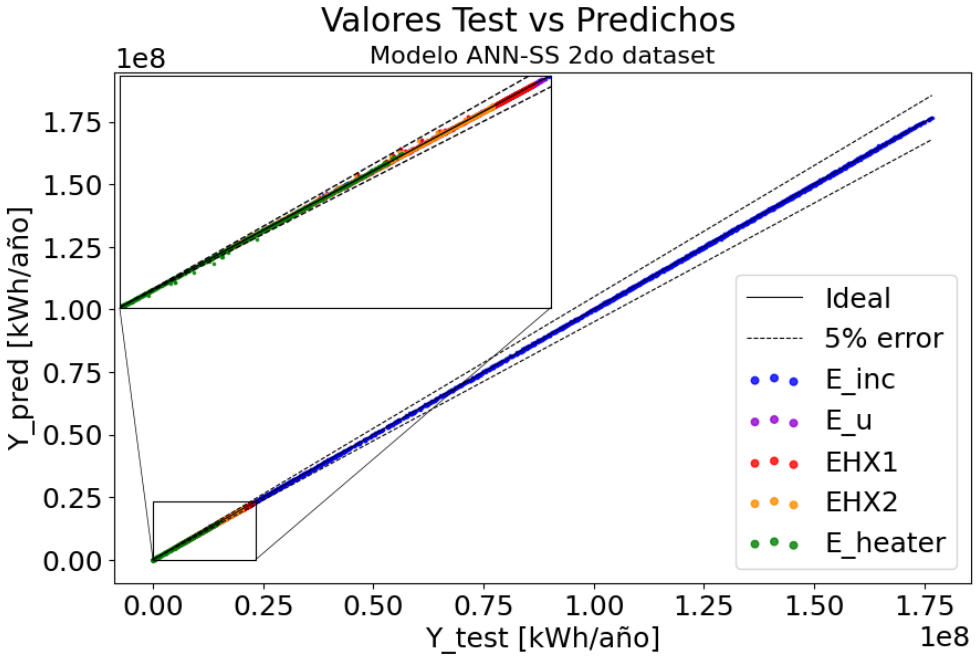
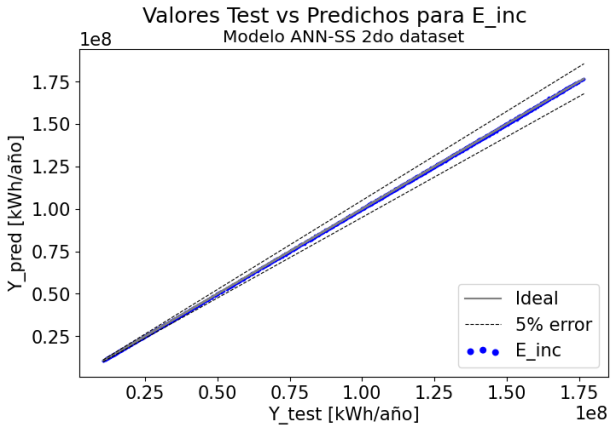


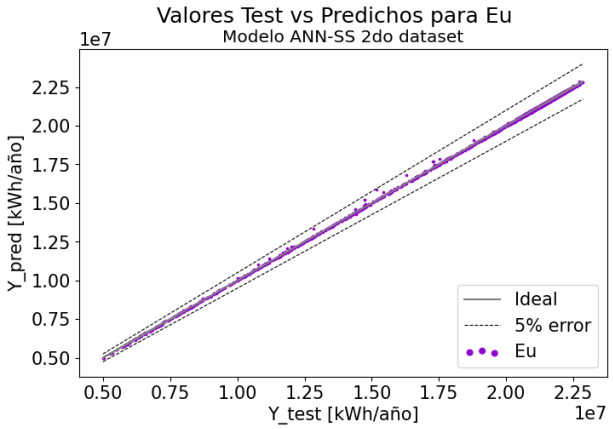
Figura 4.8: Valores test vs predichos modelo ANN-SS con segundo dataset

En la figura 4.9 también se pueden observar los resultados obtenidos, pero en este caso para cada uno de los outputs por separado. Es decir, los valores de testeo versus los predichos

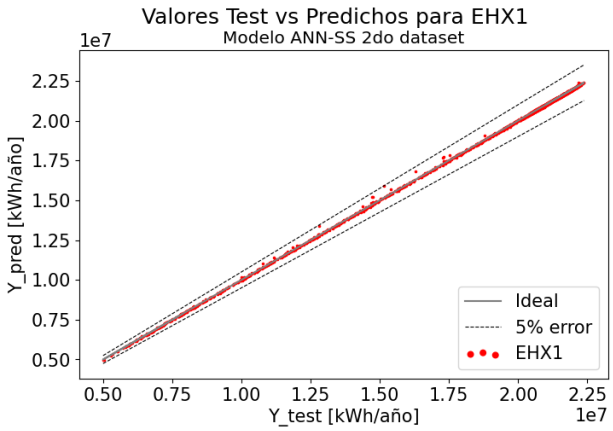
para cada output en particular desde las figuras 4.9.a a 4.9.e.



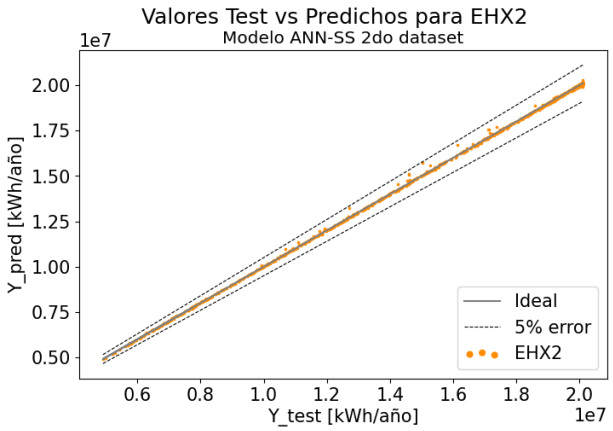
(a) Resultados E_inc



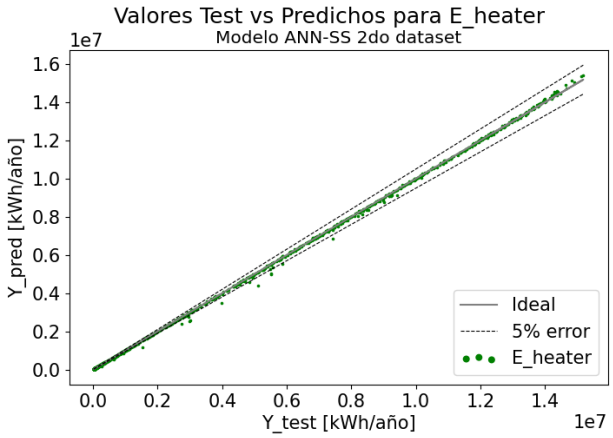
(b) Resultados Eu



(c) Resultados EHX1



(d) Resultados EHX2



(e) Resultados E_heater

Figura 4.9: Valores test vs predichos por cada output modelo ANN-SS con segundo dataset

Por último para Redes Neuronales con el segundo dataset, en la tabla 4.9 se muestran los hiperparámetros del último modelo con StandardScaler.

Tabla 4.9: Hiperparámetros para modelo ANN-SS con segundo dataset

Ctd capas ocultas	2
Nodos capa 1	256
Activación capa 1	“selu”
Nodos capa 2	512
Activación capa 2	“relu”
Learning rate	0.01

4.2.2. Árboles de Decisión con segundo dataset

Por último, para Árboles de Decisión con el segundo dataset, en primer lugar en la tabla 4.10 se presentan los datos estadísticos para cada uno de los preprocesamientos.

Tabla 4.10: Datos estadísticos Árboles de Decisión con segundo dataset

Estadística	SS	MMS	MAS	PCA3	PCA2	Nada
Mean Test Score [%]	3.58	3.54	3.36	12.32	13.82	3.63
Mean Train Score [%]	1.03	1.03	0.63	6.35	7.10	0.64
SD Test Score [%]	0.30	0.30	0.40	2.37	1.94	0.25
SD Train Score [%]	0.01	0.011	0.007	0.21	0.18	0.008

En la tabla 4.11 a continuación, se pueden observar los resultados de evaluación obtenidos. Ees decir, el error global para el conjunto de testeo como también para cada uno de los outputs en particular.

Tabla 4.11: Errores de testeo Árboles de Decisión con segundo dataset

Error	SS	MMS	MAS	PCA3	PCA2	Nada
Global [%]	3.50	3.61	3.61	12.80	14.48	3.62
E_inc [%]	0.86	0.87	0.85	3.25	3.85	0.86
Eu [%]	1.01	1.01	1.01	2.09	2.41	1.04
EHX1 [%]	0.91	0.91	0.9	1.93	2.16	0.93
EHX2 [%]	0.56	0.57	0.56	1.79	2.10	0.60
E_heater [%]	14.17	14.71	14.72	54.92	61.90	14.70

Observando las tablas anteriores, 4.10 y 4.11, nuevamente para Árboles de Decisión pero esta vez con el segundo dataset, el mejor modelo corresponde a StandardScaler. Así, en la

figura 4.10 se presenta la Learning Curve del modelo, que corresponde al Train Size versus el error para el conjunto que se utiliza de entrenamiento y de testeo en el entrenamiento.

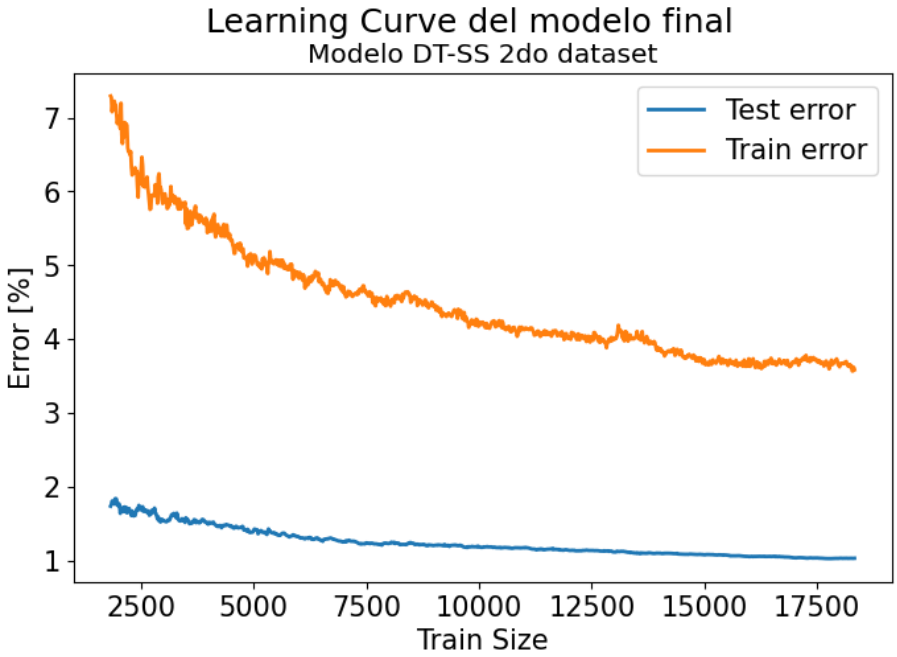


Figura 4.10: Learning Curve modelo DT-SS con segundo dataset

En la figura 4.11 se puede observar los resultados del modelo entrenado. Donde se grafican los valores de los outputs del conjunto de testeo versus los que fueron predichos por el Árbol.

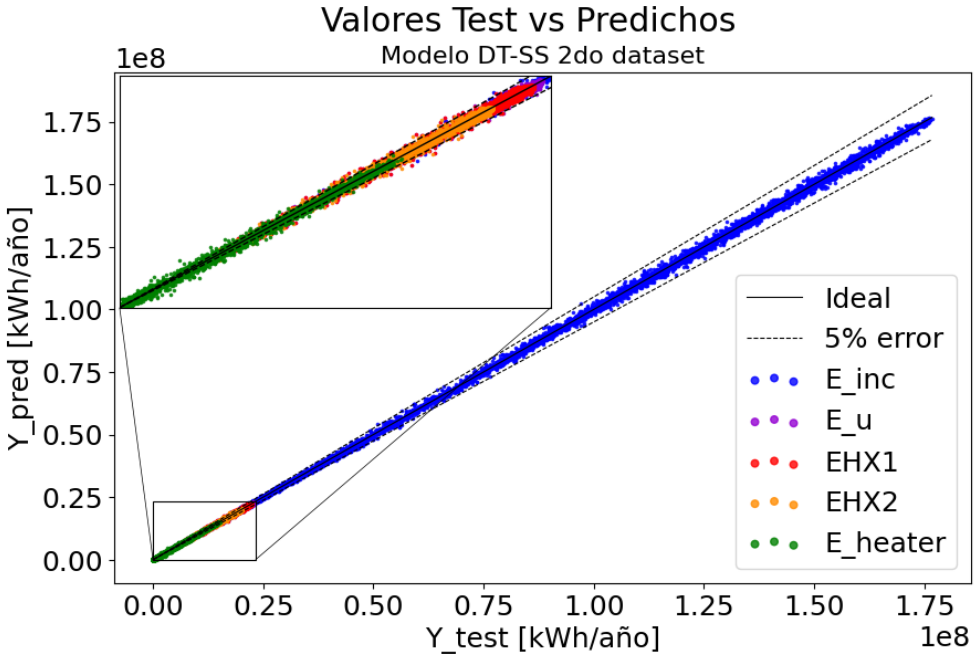
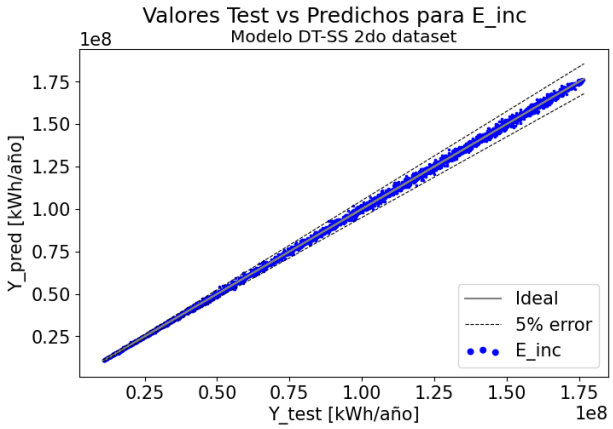


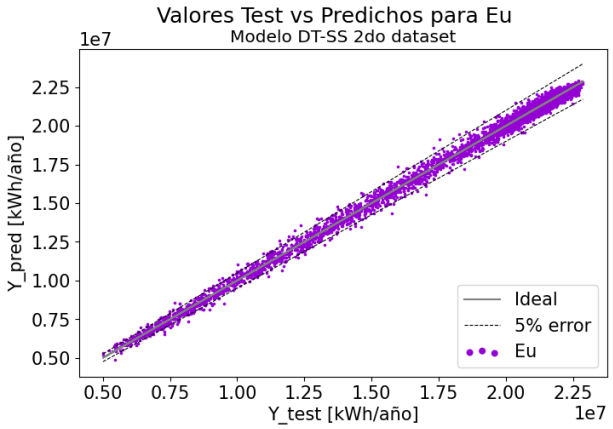
Figura 4.11: Valores test vs predichos modelo DT-SS con segundo dataset

También, en la figura 4.12 se pueden observar los resultados obtenidos, pero para cada

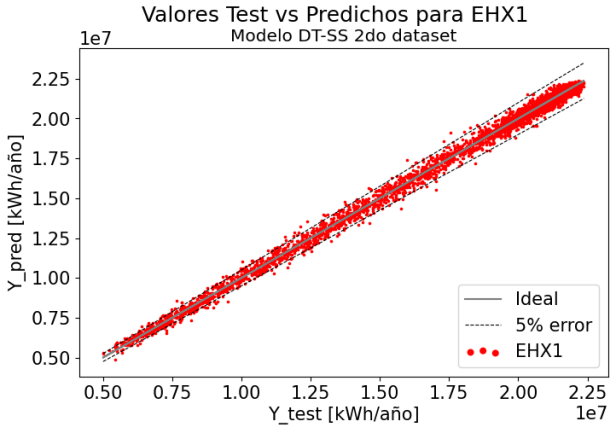
uno de los outputs en particular, desde la figura 4.12.a a 4.12.e.



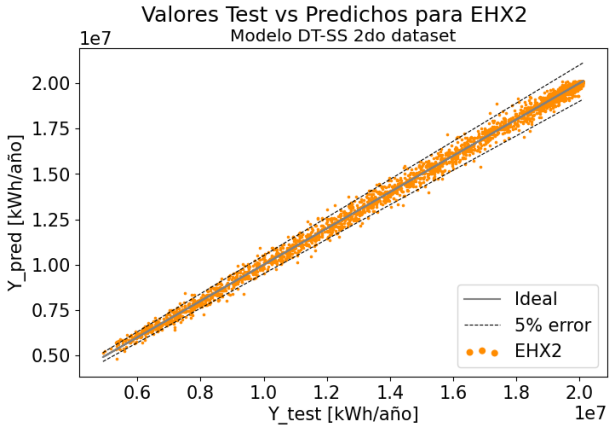
(a) Resultados E_inc



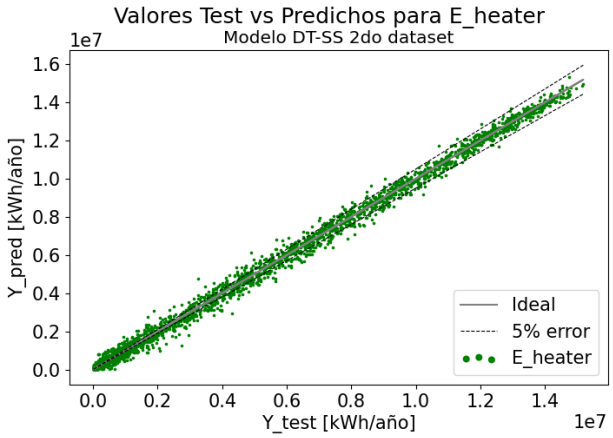
(b) Resultados Eu



(c) Resultados EHX1



(d) Resultados EHX2



(e) Resultados E_heater

Figura 4.12: Valores test vs predichos por cada output modelo DT-SS con segundo dataset

Por último, en la tabla 4.12 se pueden observar los hiperparámetros del modelo final que se obtuvo luego del tuner y torneo de hiperparámetros.

Tabla 4.12: Hiperparámetros para modelo DT-SS con segundo dataset

Criterion	"Poisson"
min_samples_split	4
Splitter	"Best"

4.3. Modelos de optimización

A continuación se presentan los resultados de las optimizaciones con los modelos de Machine Learning entrenados. Es decir, los resultados del mínimo LCoH para el caso de un objetivo, como también para el mínimo LCoH y energía de la caldera para el caso multiobjetivo, las configuraciones de las plantas solares obtenidas y por último la validación de resultados propuesta. Es decir, el error entre los resultados obtenidos y los calculados con las energías simuladas en TRNSYS-18.

4.3.1. Redes Neuronales con el primer dataset

En primer lugar, se presentan los resultados obtenidos mediante el modelo de Redes Neuronales con el primer dataset (ANN-SS 1), tanto para la optimización de un objetivo como la multiobjetivo.

4.3.1.1. Optimización de un objetivo

En la figura 4.13 se pueden observar los LCoH obtenidos en la optimización de un objetivo para cada caso estudiado.

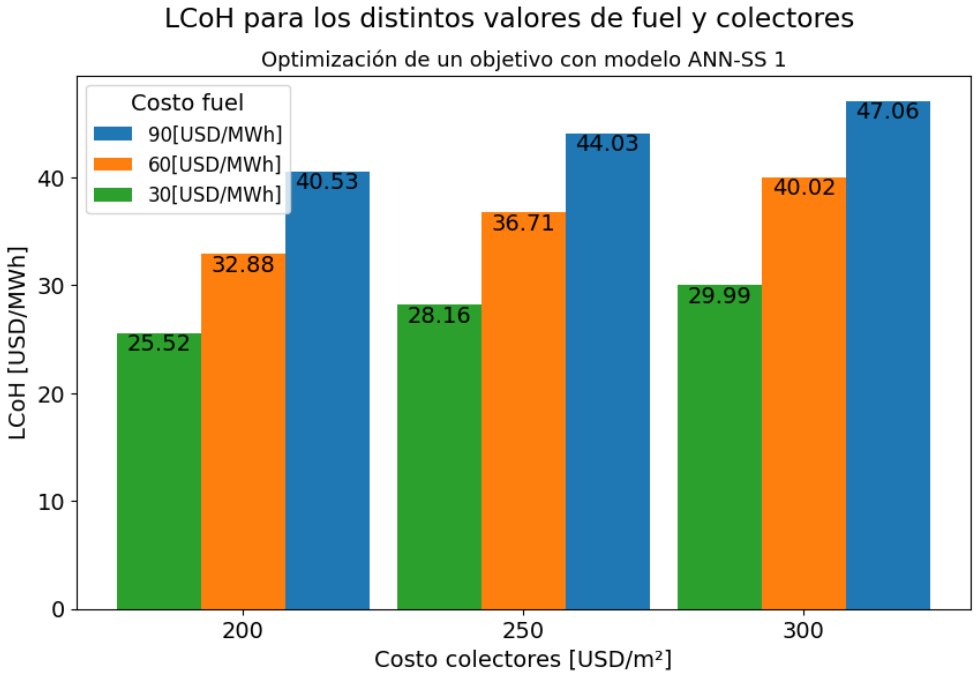


Figura 4.13: LCoH obtenidos con optimización de un objetivo ANN-SS 1

Luego, en la tabla 4.13 se pueden observar las configuraciones obtenidas. En la primera columna de costos se identifica el costo del fuel y colector respectivamente utilizados en la optimización, es decir, los valores utilizados para cada caso del análisis de sensibilidad.

Tabla 4.13: Configuraciones obtenidas para la optimización de un objetivo ANN-SS 1

Costos	Area_f [m^2]	TES_V [m^3]	beta [º]	sf_m [kg/hm^2]	ep_h [-]
30-200	9768	510	34	68	0.9
30-250	6167	215	30	57	0.9
30-300	4870	208	29	75	0.8
60-200	9802	724	31	69	0.9
60-250	9884	741	28	40	0.9
60-300	9849	766	27	49	0.9
90-200	9917	766	29	41	0.9
90-250	9854	745	26	57	0.9
90-300	9959	816	26	71	0.9

Por último, para validar los resultados se realizan simulaciones en TRNSYS-18 utilizando las configuraciones de las plantas solares. Así en la tabla 4.14 se puede observar el error porcentual entre el LCoH obtenido en las optimizaciones y el LCoH calculado con las energías obtenidas en las simulaciones mencionadas.

Tabla 4.14: Error entre LCoH obtenido y simulado para optimización de un objetivo ANN-SS 1

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	25.52	25.85	1.28
30-250	28.16	28.56	1.41
30-300	29.99	30.06	0.22
60-200	32.88	32.76	0.36
60-250	36.71	36.51	0.56
60-300	40.02	39.91	0.28
90-200	40.53	40.29	0.59
90-250	44.03	43.66	0.84
90-300	47.06	46.66	0.84

4.3.1.2. Optimización multiobjetivo

Para el modelo ANN-SS 1 utilizado en la optimización multiobjetivo en la figura 4.14 se presentan los LCoH mínimos obtenidos para el análisis de sensibilidad. Mientras que en la figura 4.15 los resultados para la energía de la caldera mínimos obtenidos en GWh/año.

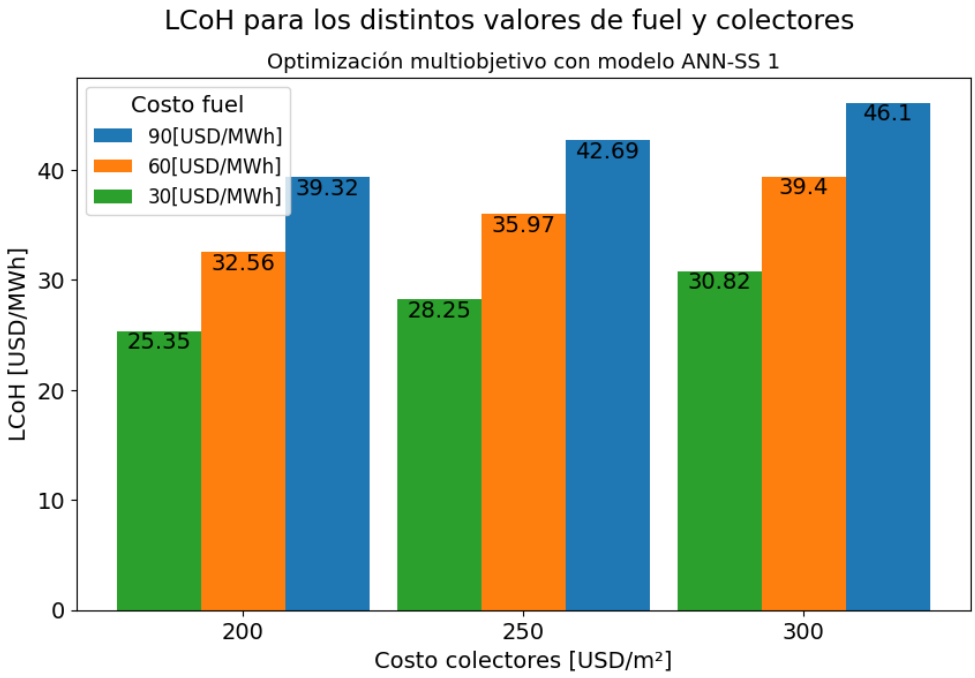


Figura 4.14: LCoH obtenidos con optimización multiobjetivo ANN-SS 1

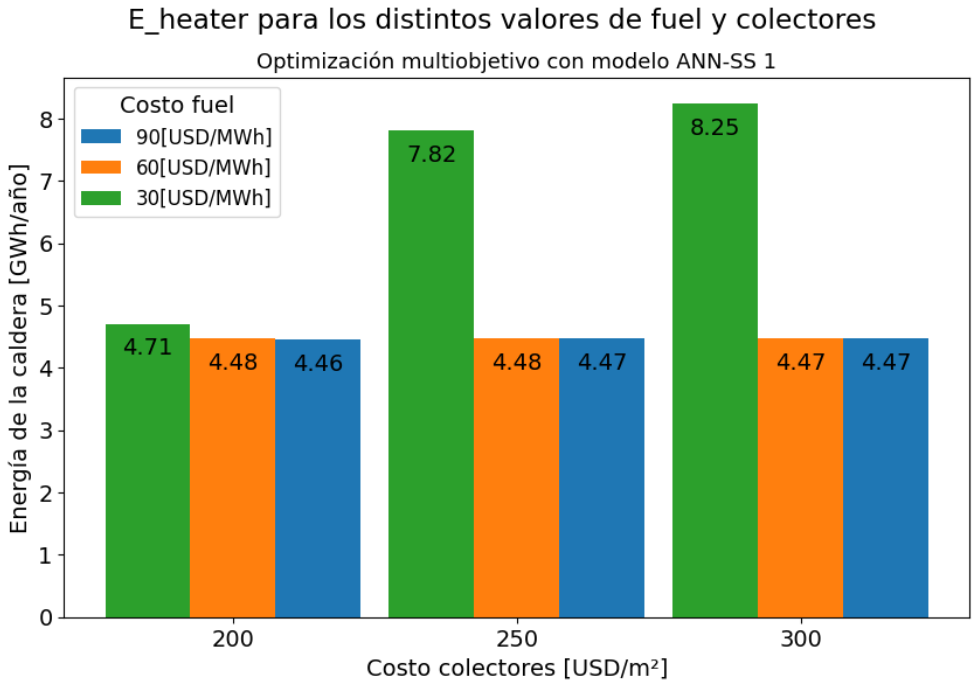


Figura 4.15: E_heater obtenidos con optimización multiobjetivo ANN-SS 1

En la tabla 4.15 a continuación, se presentan las configuraciones de la planta solar térmica obtenidas para cada uno de los casos.

Tabla 4.15: Configuraciones obtenidas para la optimización multiobjetivo ANN-SS 1

Costos	Area_f [m^2]	TES_V [m^3]	beta [º]	sf_m [kg/hm^2]	ep_h [-]
30-200	9999	687.27	32	80	0.9
30-250	7910	454.82	28	80	0.9
30-300	7632	424.67	28	80	0.9
60-200	10000	818.41	31	80	0.9
60-250	10000	817.01	31	80	0.9
60-300	10000	821.59	31	80	0.9
90-200	10000	839.79	31	80	0.9
90-250	10000	832.56	31	80	0.9
90-300	10000	830.68	31	80	0.9

Por último para el modelo ANN-SS 1, en la tabla 4.16 se pueden observar los errores para cada uno de los casos estudiados entre el LCoH obtenido y los calculados mediante nuevas simulaciones.

Tabla 4.16: Error entre LCoH obtenido y simulado para optimización multiobjetivo ANN-SS 1

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	25.35	25.26	0.35
30-250	28.25	28.36	0.41
30-300	30.82	30.94	0.39
60-200	32.56	32.50	0.19
60-250	35.97	35.91	0.18
60-300	39.40	39.34	0.16
90-200	39.32	39.22	0.26
90-250	42.69	42.60	0.20
90-300	46.10	46.01	0.21

Para finalizar con la tabla 4.17 donde se puede observar el error obtenido entre la energía de la caldera optimizado y simulado.

Tabla 4.17: Error entre E_heater obtenido y simulado para optimización multiobjetivo ANN-SS 1

Costos	E_heater [kWh/año]	E_heater simulado [kWh/año]	Error [%]
30-200	4709457.5	4650117.24	1.26
30-250	7820685.5	7891344.39	0.90
30-300	8248605.5	8327604.21	0.96
60-200	4475711.0	4452018.35	0.53
60-250	4476506.0	4452797.77	0.53
60-300	4473280.0	4448898.97	0.55
90-200	4461215.0	4436977.95	0.54
90-250	4465799.0	4441552.44	0.54
90-300	4467019.0	4442701.65	0.54

4.3.2. Árboles de Decisión con el primer dataset

A continuación se presentan los resultados obtenidos mediante el modelo de Árboles de Decisión con el primer dataset (DT-SS 1), para ambas optimizaciones.

4.3.2.1. Optimización de un objetivo

En la figura 4.16 se pueden observar los LCoH mínimos obtenidos para cada caso estudiado.

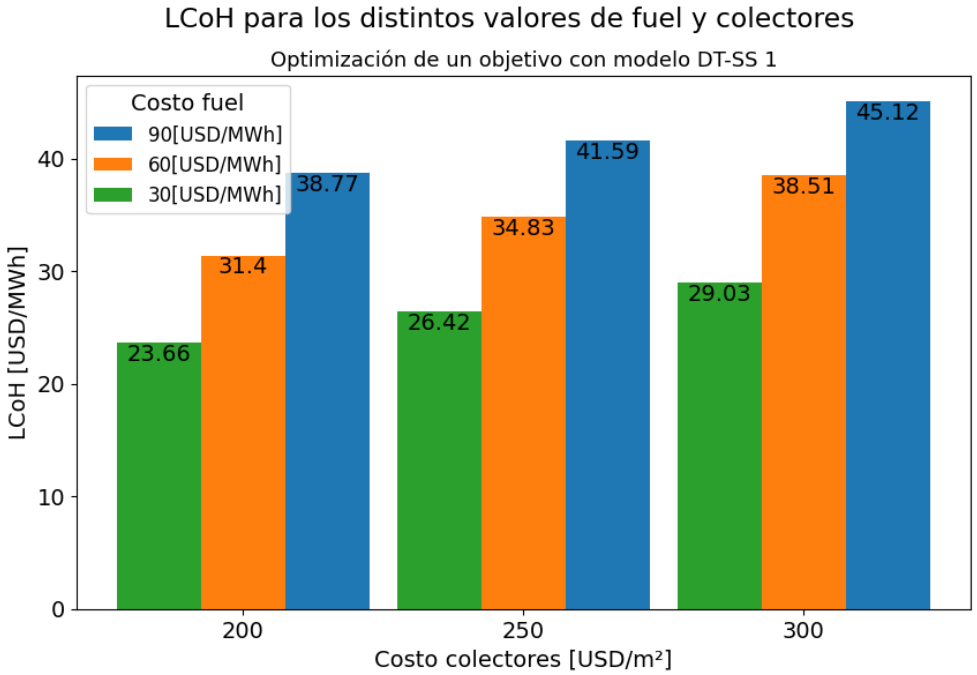


Figura 4.16: LCoH obtenidos con optimización de un objetivo DT-SS 1

Mientras que en la tabla 4.18 se presentan las configuraciones de la planta solar obtenidas.

Tabla 4.18: Configuraciones obtenidas para la optimización de un objetivo DT-SS 1

Costos	Area_f [m2]	TES_V [m3]	beta [º]	sf_m [kg/hm2]	ep_h [-]
30-200	8567	240	15	44	0.9
30-250	8578	214	10	45	0.9
30-300	5512	233	31	78	0.8
60-200	9564	663	29	41	0.9
60-250	9599	679	33	44	0.9
60-300	9677	716	34	40	0.9
90-200	9684	707	35	44	0.9
90-250	9538	672	30	43	0.9
90-300	9704	663	27	50	0.9

Finalmente, en la tabla 4.19 se puede observar el error porcentual entre el LCoH optimizado con el LCoH calculado con las energías obtenidas mediante nuevas simulaciones.

Tabla 4.19: Error entre LCoH obtenido y simulado para optimización de un objetivo DT-SS 1

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	23.66	28.66	21.12
30-250	26.42	32.23	21.98
30-300	29.03	30.23	4.13
60-200	31.40	33.71	7.35
60-250	34.83	37.07	6.43
60-300	38.51	40.48	5.11
90-200	38.77	41.82	7.86
90-250	41.59	45.07	8.37
90-300	45.12	47.74	5.81

4.3.2.2. Optimización multiobjetivo

Para el modelo DT-SS 1, pero esta vez utilizado en la optimización multiobjetivo, en la figura 4.17 se pueden observar los LCoH mínimos obtenidos.

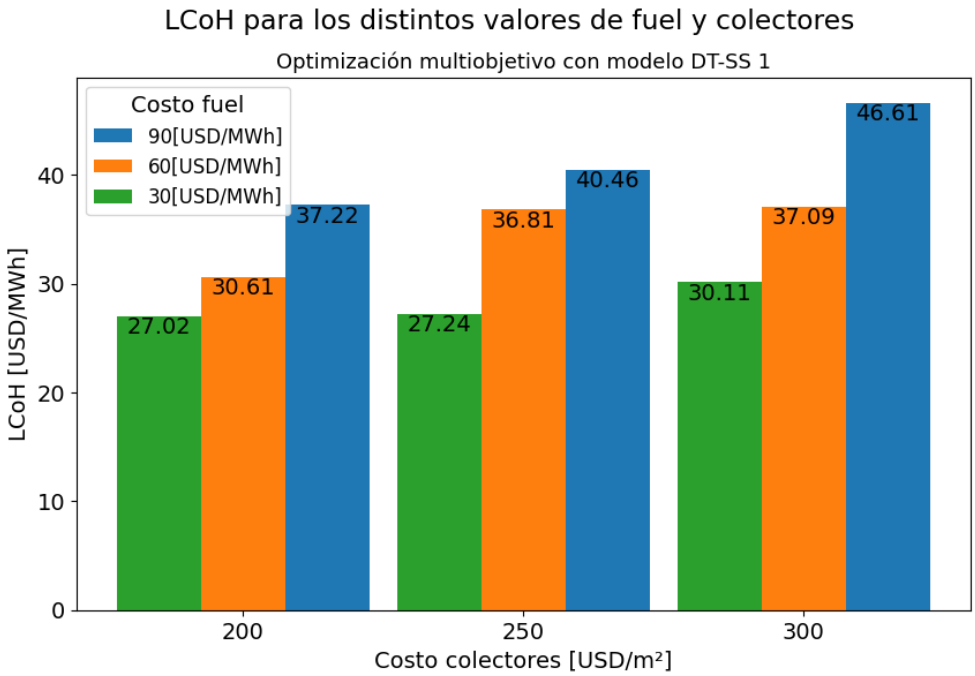


Figura 4.17: LCoH obtenidos con optimización multiobjetivo DT-SS 1

En la figura 4.18 a continuación, se pueden observar la energía de la caldera obtenidos también para cada caso.

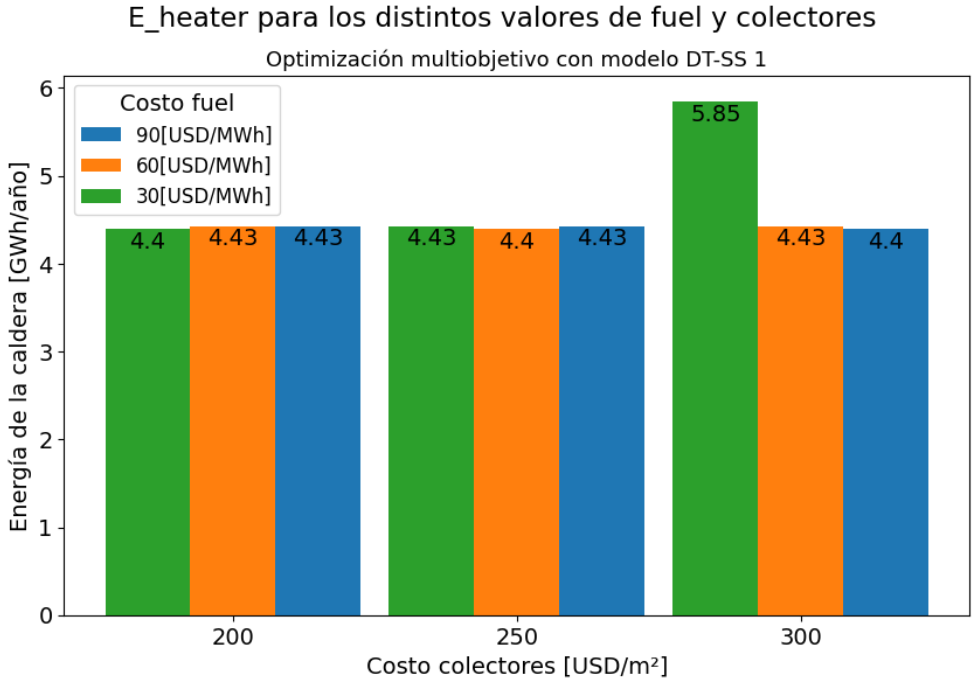


Figura 4.18: E_heater obtenidos con optimización multiobjetivo DT-SS 1

Además, en la tabla 4.20 se pueden observar las configuraciones de la planta solar obtenidas en la optimización realizada.

Tabla 4.20: Configuraciones obtenidas para la optimización multiobjetivo DT-SS 1

Costos	Area_f [m2]	TES_V [m3]	beta [º]	sf_m [kg/hm2]	ep_h [-]
30-200	9500	1100.02	27	79	0.9
30-250	9500	650.04	30	77	0.9
30-300	8500	585.18	28	76	0.9
60-200	9500	650.05	33	68	0.9
60-250	9500	1100.03	28	79	0.9
60-300	9500	650.00	33	80	0.9
90-200	9500	650.07	33	76	0.9
90-250	9500	650.02	31	77	0.9
90-300	9500	1100.00	29	76	0.9

Y por último para Árboles de Decisión con el primer dataset, en las tabla 4.21 y 4.22 se pueden observar los errores entre los resultados obtenidos y los calculados con nuevas simulaciones.

Tabla 4.21: Error entre LCoH obtenido y simulado para optimización multiobjetivo DT-SS 1

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	27.02	27.86	3.14
30-250	27.24	28.57	4.90
30-300	30.11	31.44	4.42
60-200	30.61	33.59	9.76
60-250	36.81	38.48	4.55
60-300	37.09	39.88	7.52
90-200	37.22	41.50	11.49
90-250	40.46	44.51	9.99
90-300	46.61	49.20	5.55

Tabla 4.22: Error entre E_heater obtenido y simulado para optimización multiobjetivo DT-SS 1

Costos	E_heater [kWh/año]	E_heater simulado [kWh/año]	Error [%]
30-200	4395131.07	4964209.71	12.95
30-250	4431703.36	5324016.49	20.13
30-300	5850198.90	6740536.53	15.22
60-200	4431703.36	5432733.02	22.59
60-250	4395131.07	4954736.43	12.73
60-300	4431703.36	5365747.90	21.08
90-200	4431703.36	5387874.09	21.58
90-250	4431703.36	5336423.92	20.41
90-300	4395131.07	4973136.65	13.15

4.3.3. Redes Neuronales con el segundo dataset

Para Redes Neuronales con el segundo dataset (ANN-SS 2), a continuación se presentan los resultados para las dos optimizaciones realizadas.

4.3.3.1. Optimización de un objetivo

En la figura 4.19 se presentan los resultados para los LCoH obtenidos.

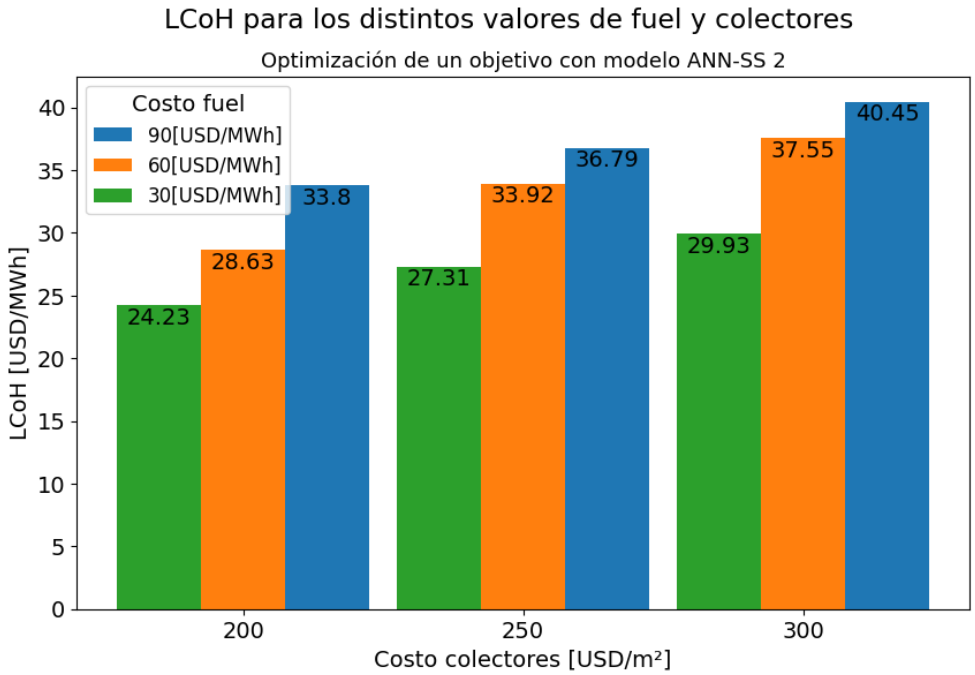


Figura 4.19: LCoH obtenidos con optimización de un objetivo ANN-SS 2

Y en la tabla 4.23 las configuraciones de la planta solar obtenidas para cada caso.

Tabla 4.23: Configuraciones obtenidas para la optimización de un objetivo ANN-SS 2

Costos	Area_f [m^2]	TES_V [m^3]	beta [º]	sf_m [kg/hm^2]	ep_h [-]
30-200	10294	524.99	28	71	0.9
30-250	9789	509.03	28	99	0.9
30-300	5197	259.85	21	97	0.9
60-200	12733	763.98	41	88	0.9
60-250	11517	702.54	34	53	0.9
60-300	11204	506.20	34	52	0.8
90-200	11893	606.54	38	42	0.9
90-250	12060	615.06	37	69	0.9
90-300	12929	827.46	41	94	0.9

Por último, en la tabla 4.24 se pueden observar los resultados del error entre el LCoH obtenido en la optimización de un objetivo, y la calculada mediante las energías obtenidas en nuevas simulaciones.

Tabla 4.24: Error entre LCoH obtenido y simulado para optimización de un objetivo ANN-SS 2

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	24.23	26.23	8.24
30-250	27.31	29.22	6.99
30-300	29.93	29.99	0.22
60-200	28.63	29.76	3.94
60-250	33.92	34.69	2.29
60-300	37.55	41.99	11.82
90-200	33.80	39.33	16.36
90-250	36.79	42.37	15.18
90-300	40.45	41.26	2.00

4.3.3.2. Optimización multiobjetivo

Para el modelo ANN-SS 2, en la figura 4.20 y 4.21 se tienen los resultados obtenidos para el LCoH y energía de la caldera respectivamente para el análisis de sensibilidad realizado para la optimización multiobjetivo.

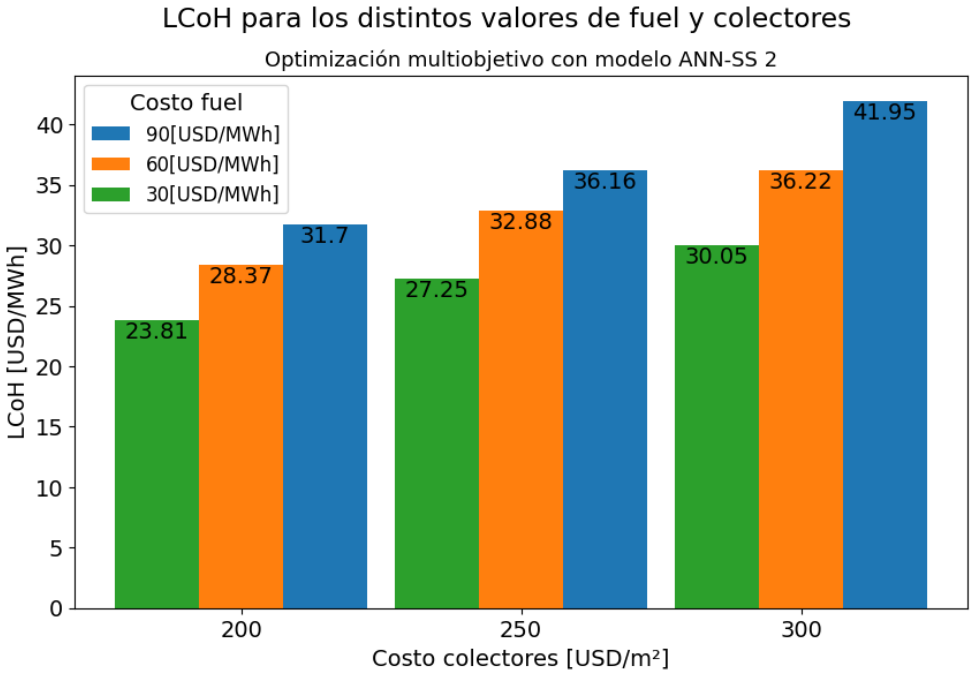


Figura 4.20: LCoH obtenidos con optimización multiobjetivo ANN-SS 2

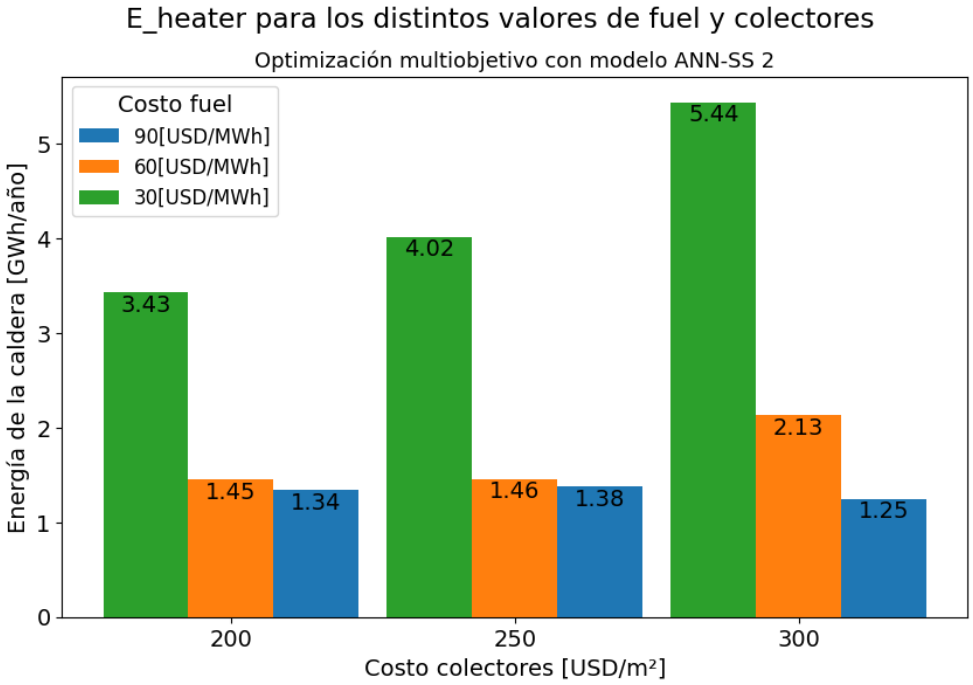


Figura 4.21: E_heater obtenidos con optimización multiobjetivo ANN-SS 2

Mientras que en la tabla 4.25 se pueden observar las configuraciones de la planta solar obtenidas en la optimización.

Tabla 4.25: Configuraciones obtenidas para la optimización multiobjetivo ANN-SS 2

Costos	Area_f [m^2]	TES_V [m^3]	beta [º]	sf_m [kg/hm^2]	ep_h [-]
30-200	10925	546.29	33	100	0.9
30-250	10375	518.77	32	100	0.9
30-300	9177	458.86	28	100	0.9
60-200	13479	815.09	46	100	0.9
60-250	13404	814.22	45	100	0.9
60-300	12473	623.66	39	100	0.9
90-200	13485	1059.30	45	100	0.9
90-250	13417	1024.36	44	100	0.9
90-300	13675	1218.95	46	100	0.9

Por último, en las tabla 4.26 y 4.27 se pueden observar los errores entre los LCoH y energía de la caldera obtenidos y los calculados con nuevas simulaciones respectivamente.

Tabla 4.26: Error entre LCoH obtenido y simulado para optimización multiobjetivo ANN-SS 2

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	23.81	26.17	9.93
30-250	27.25	29.57	8.52
30-300	30.05	32.18	7.10
60-200	28.37	29.86	5.26
60-250	32.88	34.28	4.25
60-300	36.22	40.63	12.18
90-200	31.70	32.06	1.12
90-250	36.16	36.50	0.93
90-300	41.95	41.84	0.26

Tabla 4.27: Error entre E_heater obtenido y simulado para optimización multiobjetivo ANN-SS 2

Costos	E_heater [kWh/año]	E_heater simulado [kWh/año]	Error [%]
30-200	3429633.5	5059107.97	47.51
30-250	4018770.5	5590476.59	39.11
30-300	5440870.5	6887342.01	26.58
60-200	1454578.4	1984647.97	36.44
60-250	1461140.2	1968984.37	34.76
60-300	2128475.8	3638181.93	70.92
90-200	1339955.8	1439951.69	7.46
90-250	1383626.4	1483899.73	7.24
90-300	1245227.1	1240700.99	0.36

4.3.4. Árboles de Decisión con el segundo dataset

Por último, se obtienen los resultados para el modelo de Árboles de Decisión con el segundo dataset (DT-SS 2).

4.3.4.1. Optimización de un objetivo

En la figura 4.22 se pueden observar los LCoH obtenidos para cada caso estudiado.

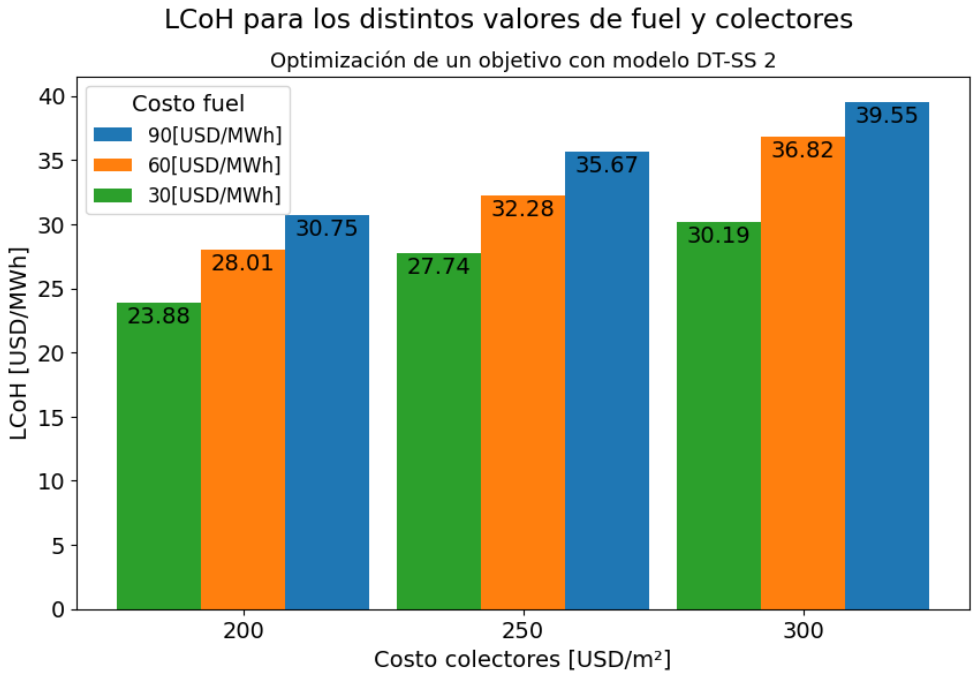


Figura 4.22: LCoH obtenidos con optimización de un objetivo DT-SS 2

Y en la tabla 4.28 las configuraciones de la planta solar obtenidas en cada uno de los casos.

Tabla 4.28: Configuraciones obtenidas para la optimización de un objetivo DT-SS 2

Costos	Area_f [m2]	TES_V [m3]	beta [º]	sf_m [kg/hm2]	ep_h [-]
30-200	11103	555.15	38	52	0.9
30-250	10644	553.49	27	98	0.8
30-300	6830	402.97	13	93	0.9
60-200	12355	679.53	40	73	0.9
60-250	11430	640.08	31	90	0.9
60-300	12265	625.52	42	94	0.8
90-200	14025	869.55	50	80	0.9
90-250	12366	704.86	35	40	0.9
90-300	13162	776.56	33	92	0.9

Finalmente, en la tabla 4.29 se pueden observar la validación de resultados para la optimización realizada. Es decir, se presentan los errores entre el LCoH obtenido y el calculado con nuevas simulaciones.

Tabla 4.29: Error emtre LCoH obtenido y simulado para optimización de un objetivo DT-SS 2

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	23.88	26.48	10.91
30-250	27.74	29.99	8.11
30-300	30.19	31.24	3.48
60-200	28.01	30.88	10.25
60-250	32.28	35.39	9.63
60-300	36.82	41.04	11.46
90-200	30.75	33.49	8.92
90-250	35.67	39.53	10.82
90-300	39.55	42.29	6.92

4.3.4.2. Optimización multiobjetivo

En la figura 4.23 se tienen los resultados del LCoH obtenidos para cada uno de los casos del análisis de sensibilidad para la optimización multiobjetivo. Mientras que en la figura 4.24 se tienen las energías de la caldera obtenidos para cada uno de los casos estudiados.

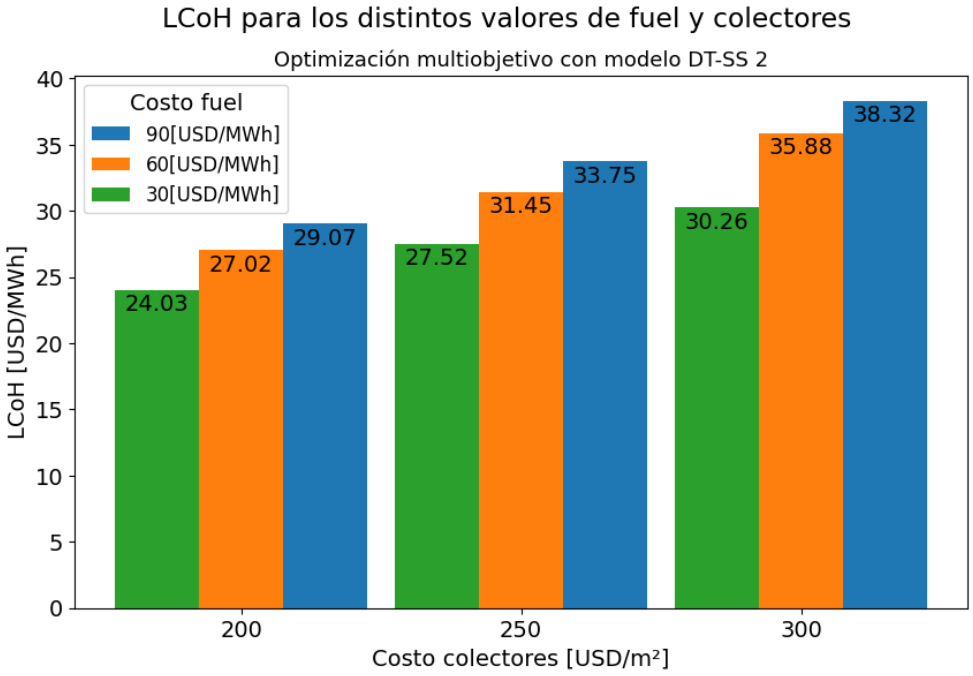


Figura 4.23: LCoH obtenidos con optimización multiobjetivo DT-SS 2

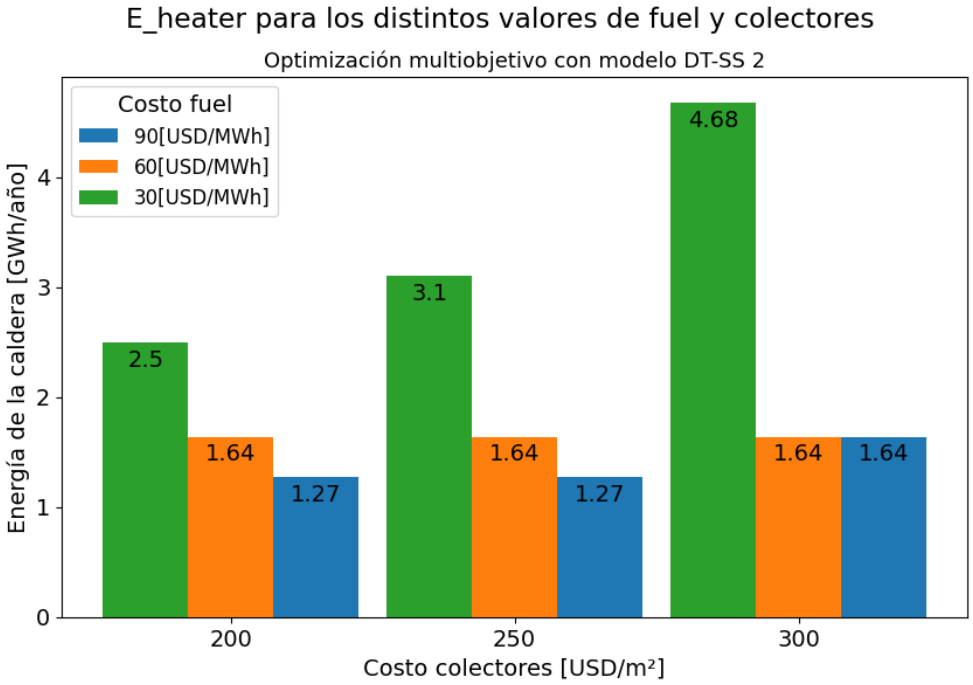


Figura 4.24: E_heater obtenidos con optimización multiobjetivo DT-SS 2

En la tabla 4.30 se presentan los resultados de las configuraciones de la planta solar obtenidas para cada uno de los casos.

Tabla 4.30: Configuraciones obtenidas para la optimización multiobjetivo DT-SS 2

Costos	Area_f [m2]	TES_V [m3]	beta [º]	sf_m [kg/hm2]	ep_h [-]
30-200	11905	595.26	40	82	0.9
30-250	11187	559.37	40	99	0.9
30-300	9744	487.21	32	98	0.8
60-200	12985	649.26	32	91	0.9
60-250	12984	649.18	37	93	0.9
60-300	12983	649.15	38	80	0.8
90-200	13714	685.71	48	83	0.9
90-250	13715	685.73	51	99	0.9
90-300	12984	649.18	35	83	0.9

Y por último, en las tablas 4.31 y 4.32 se pueden observar los resultados obtenidos para la validación de resultados correspondiente. En la primera tabla se muestran los errores entre el LCoH obtenido y calculado con las nuevas simulaciones, mientras que en la segunda el error entre las energías de la caldera.

Tabla 4.31: Error entre LCoH obtenido y simulado para optimización multiobjetivo DT-SS 2

Costos	LCoH [USD/MWh]	LCoH simulado [USD/MWh]	Error [%]
30-200	24.03	26.60	10.68
30-250	27.52	30.10	9.38
30-300	30.26	32.77	8.30
60-200	27.02	31.88	17.97
60-250	31.45	36.15	14.94
60-300	35.88	41.07	14.46
90-200	29.07	37.07	27.51
90-250	33.75	42.70	26.51
90-300	38.32	45.52	18.79

Tabla 4.32: Error entre E_heater obtenido y simulado para optimización multiobjetivo DT-SS 2

Costos	E_heater [kWh/año]	E_heater simulado [kWh/año]	Error [%]
30-200	2498242.36	4221563.12	68.98
30-250	3098505.46	4834076.62	56.01
30-300	4678983.34	6373446.74	36.21
60-200	1635905.20	3264977.84	99.58
60-250	1635905.20	3213802.76	96.45
60-300	1635905.20	3378667.50	106.53
90-200	1271198.71	3058238.85	140.58
90-250	1271198.71	3270500.33	157.27
90-300	1635905.20	3247353.21	98.50

Capítulo 5

Análisis de resultados

A continuación se detallan los análisis y discusión en profundidad de los resultados obtenidos tanto para los modelos de Machine Learning, optimizaciones realizadas y la validación de los datos obtenidos.

5.1. Modelos de Machine Learning

Lo primero que se puede notar de los resultados obtenidos es que para todos los modelos estudiados, tanto para el primer y segundo dataset, el mejor preprocesamiento obtenido corresponde a Standard Scaler. Pero también si se observan las tablas 4.2, 4.5, 4.8 y 4.11, para los modelos entrenados y testeados los preprocesamientos MinMax Scaler y MaxAbs Scaler no tienen errores muy diferentes a los de Standard Scaler, dando a notar que la normalización de datos funciona de mejor manera para los casos estudiados que el Análisis de Componentes Principales (PCA) y sin preprocesamiento. Es posible que el PCA no funcione de buena manera debido a la naturaleza de los datos y no se puedan hacer buenas regresiones entre ellos. Por otro lado, el error alto en la mayoría de los casos de no usar preprocesamiento indica que, como se mencionó en capitulos anteriores, es necesario realizar preprocesamientos para obtener buenos modelos de Machine Learning.

5.1.1. Modelos con el primer dataset

En primer lugar, observando la tabla 4.1 para el análisis estadístico de Redes Neuronales, se puede notar que el preprocesamiento con Standard Scaler es el modelo con menores porcentajes, con un 0.16 % y 0.19 % de promedio de errores y 0.03 % de desviación estándar. Lo que significa que al re-entrenar el modelo todos convergen a valores parecidos de pérdidas. Mientras que los otros modelos comienzan a divergir, dando a entender cierta dependencia de la inicialización de pesos de la Red Neuronal. Luego al continuar los entrenamientos en la tabla 4.2 se puede notar que el modelo con menor error de testeo corresponde a MaxAbs Scaler con 0.09 % seguido por Standard Scaler con una diferencia de 0.03 %. Esto también se puede observar en la figura 4.1 versus A.2. En estas se observa que Standard Scaler converge en un poco más de 400 épocas, mientras que MaxAbs Scaler en 1600 épocas, lo que aumenta

el tiempo de entrenamiento de manera significativa. Por estas dos cosas, Standard Scaler es considerado el modelo de Redes Neuronales entrenado con el primer dataset con mejores resultados dentro de los preprocesamientos estudiados. En las figuras 4.2 y 4.3 se puede observar de manera gráfica lo anteriormente mencionado. En la primera figura se observan todos los outputs mientras que en la segunda para cada uno por separado. Se puede notar que la mayoría de los resultados se encuentran dentro de la línea de identidad ideal, y los valores que divergen de esta se encuentran dentro del rango seleccionado del 5 % de error validando de mejor manera que el modelo seleccionado tiene una buena capacidad de predicción.

Para el caso de Árboles de decisión, en la tabla 4.4 se puede notar que en general para todos los preprocesamientos los promedios y desviaciones estándar el último valor de error de las validaciones cruzadas no superan el 5 % y son relativamente bajas, nuevamente divergiendo de mayor manera los con PCA. Esto, a diferencia de Redes Neuronales, se puede deber a que al entrenar los Árboles estos no se restringen pudiendo llegar a la profundidad necesaria para la convergencia, mientras que Redes se restringen por la cantidad de épocas. En la tabla 4.5 se puede notar que al testear el Árbol, el que tiene un menor error corresponde a Standard Scaler con un 0.17 %, siendo nuevamente el preprocesamiento con mejor desempeño, pero con resultados similares a MaxAbs Scaler, MinMax Scaler y sin ningún tipo de preprocesamiento. Pero si se observan la figuras 4.5 y 4.6 en su mayoría los resultados test versus los predichos por el Árbol con Standard Scaler, se encuentran dentro de la línea de la identidad ideal, y los que divergen dentro del margen de error del 5 %. Mientras que en las figuras D.2, D.1 y D.5, se puede notar que para MinMax Scaler, MaxAbs Scaler y sin preprocesamiento si se tienen valores que divergen fuera del rango. Disminuyendo la capacidad de predicción que se requiere de los modelos para realizar la posterior optimización. Por esto, nuevamente Standard Scaler se considera como el mejor preprocesamiento.

5.1.2. Modelos el con segundo dataset

Para Redes Neuronales con el segundo dataset, en la tabla 4.7 se puede observar que las diferencias entre las pérdidas, tanto el promedio como la desviación estándar son mucho mayores que para el primer dataset, lo que significa que para este caso los modelos dependen de mayor manera de la inicialización de pesos. Se puede notar también que nuevamente el que tiene menores diferencias corresponde a Standard Scaler. También en la tabla 4.8 se pueden observar los errores obtenidos luego de el testeo de los últimos modelos completamente entrenados. Aquí se ve que el que tiene menor error corresponde a Standard Scaler con un 3.5 %. Si se observa completamente la tabla se puede ver que los errores para las primeros cuatro outputs los errores son considerablemente bajos, pero aumenta en gran medida para la energía de la caldera, siendo el output con error mas alto de 1.44 % en comparación a 0.18 % que sigue de la energía incidente. De hecho para el caso sin preprocesamiento la Red definitivamente no es capaz de obtener los resultados con un error del 100 %. Dando a entender que la Red Neuronal tiene dificultades al momento de predecir este output en particular en comparación a los otros, lo que se puede deber a que es la energía con menos relación a las otras y es más difícil realizar la regresión de resultados. Esto también se puede observar en los gráficos de valores test versus predichos en las figuras 4.8 y 4.9. Nue-

vamente que la capacidad de predicción es buena, estando la mayoría de los valores dentro de la línea ideal, pero para la energía de la caldera hay valores fuera del rango del 5 % de error.

Observando la convergencia de los modelos, para el caso de Standard Scaler en la figura 4.7 se puede observar que en el entrenamiento las pérdidas convergen de buena manera, en aproximadamente 500 épocas. Mientras que en las figuras de anexos E.1 a E.5 para las normalizaciones MinMax Scaler y MaxAbs Scaler también tiene buena convergencia, pero en una cantidad de épocas considerablemente mayor, el doble para el caso de MaxAbs Scaler, es decir 1200 épocas aproximadamente y 2500 épocas para MinMax Scaler aproximadamente, lo que al igual de lo que se mencionó para el primer dataset, esto aumenta considerablemente el tiempo de entrenamiento. Así, nuevamente por todo lo mencionado, Standard Scaler es considerado el mejor preprocesamiento para este caso.

Para Árboles de Decisión entrenados con el segundo dataset, en la tabla 4.10 se puede observar que el promedio de los últimos errores de las validaciones cruzadas es mayor a los obtenidos con el primer dataset. Pero la desviación estándar son mucho menores, por lo que en este caso los modelos convergen al error obtenido. Lo que se puede observar en la tabla 4.11, donde los errores de testeo son muy parecidos al promedio obtenido en el análisis estadístico. Nuevamente la energía de la caldera es el que tiene un mayor error en comparación a los demás valores, al igual que para Redes Neuronales. De hecho este valor es el que termina aumentando el error global ya que los demás son los suficientemente bajos. En el caso de Standard Scaler, que corresponde al mejor modelo dentro de los otros preprocesamientos el error para la energía de la caldera aumenta considerablemente en comparación de los otros outputs con un 14.14 % de error en comparación a 1.01 % del que lo sigue.

Lo mencionado se puede observar de manera más gráfica en la figura 4.11, donde a diferencia de los modelos anteriores mencionados, los valores en su mayoría se encuentran fuera de la línea de la identidad ideal, teniendo que el modelo funciona de peor manera para la predicción de datos, además en general este presenta muchos más datos que salen del rango del 5 % de error. Lo que se puede observar de mejor manera para cada uno de los outputs por separado en la figura 4.12. Donde, consistente con el error obtenido de testeo, el que más diverge de la linea de identidad y error del 5 % corresponde a la energía de la caldera.

Para ambos modelos, Redes Neuronales y Árboles de Decisión, se pudo observar que los errores de testeo son mayores a los obtenidos en el primer dataset. Esto se puede deber a que para estos modelos es más difícil hacer una regresión debido a la naturaleza de los datos. Ya que para el segundo dataset los datos son heterogéneos, es decir, varían más entre sí que los datos del primer dataset que son equiespaciados.

5.2. Modelos de Optimización

Considerando todas las optimizaciones realizadas, lo primero que se puede notar observando las figuras 4.13, 4.16, 4.14, 4.17, 4.19, 4.22, 4.20 y 4.23 es que para los dos tipos de

optimización y en ambos datasets, el LCoH va en aumento a medida que aumentan los precios del combustible y de los colectores. Lo que tiene sentido considerando que al aumentar el valor del combustible aumentan los precios de operación, y al aumentar los precios de los colectores aumenta la inversión inicial. Y así, al aumentar uno o el otro, el LCoH siempre va a aumentar. Es decir, el LCoH es muy sensible a los precios de mercado de estas variables.

Mientras que, para el caso de las optimizaciones multiobjetivos, en las figuras 4.15, 4.18, 4.21 y 4.24 se puede observar que la energía de la caldera en su mayoría va disminuyendo, es decir, aumenta considerablemente para precios de combustible bajos y disminuye para precios altos. Lo que tiene sentido, considerando que también se busca minimizar el LCoH de la planta. Mientras la energía de la caldera es más cara, menos se utiliza.

Centrando la discusión sólo a los resultados de las optimizaciones utilizando modelos de Machine Learning entrenados con el primer dataset, lo primero que se puede notar y más importante es que tienen mejor desempeño que utilizando los modelos con el segundo dataset. Esto se puede notar observando las tablas de validación de resultados correspondientes. En primer lugar, en el caso de Redes Neuronales esto se puede observar mediante las tablas 4.14, 4.16 y 4.17, donde ni para el LCoH ni para la energía de la caldera el error con respecto a los datos simulados supera el 2 %. Otro punto importante de mencionar corresponde a la capacidad de interpolación que tiene el modelo. Si se observa la tabla 4.13 y 4.15, algunos de los resultados corresponden a áreas, volúmenes, etc. que no pertenecen a sus conjuntos dentro del dataset. Y a pesar de esto, el modelo es capaz de predecir las energías con un bajo error y buena capacidad de predicción.

Para el caso de las optimizaciones con Árboles de Decisión entrenados con el primer dataset, en las tablas 4.19, 4.21 y 4.22 se puede notar que los errores aumentan considerablemente, llegando hasta los 22.59 % de error en el caso de la energía de la caldera. Esto muestra que este modelo no es lo suficientemente bueno en comparación a las Redes Neuronales. Además, si se observan las tablas de las configuraciones 4.18 y 4.30 los resultados son muy parecidos entre sí. Lo que da a entender que no sólo existe una mala interpolación de resultados, si no que el modelo entrega resultados muy parecidos entre sí para diferentes costos de fuel y paneles. Lo que se puede deber a que Árboles de Decisión entrega resultados discretos en comparación a Redes Neuronales. Con esto se quiere decir que Árboles de Decisión entrega resultados que estén en las hojas terminales de este, y quizás el problema planteado es muy complejo y no se obtiene una cantidad de resultados suficientes para realizar una buena interpolación. Para el caso de la optimización multiobjetivo esto se puede observar mejor en las figuras J.1 a J.9 en anexo. Donde se puede notar los datos obtenidos no son capaces de formar la curva del frente de Pareto, si no que se obtienen múltiples veces los mismos resultados.

Haciendo la comparación entre los resultados obtenidos para las optimizaciones de un objetivo y multiobjetivo, si se observan nuevamente las imágenes 4.13 y 4.14 se puede notar que los LCoH obtenidos no varían mucho entre sí. Esto puede ser debido a que la energía de la caldera en comparación a las otras energías que se obtienen de output es muy pequeña.

Es decir, quizás la energía de la caldera no tiene una incidencia demasiado grande dentro del cálculo del LCoH para la planta solar utilizada en específico.

Por último, si se observan las tablas mencionadas de las configuraciones de la planta solar obtenidas para ambos modelos de Machine Learning para ambas optimizaciones, es decir 4.13, 4.15, 4.18 y 4.20, se puede notar que existe una saturación del área de colectores. Es decir, llega un punto en que el máximo del dataset con el que fue entrenado el modelo de Machine Learning corresponde a lo óptimo. Lo que significaría que para obtener mejores resultados, que quizás incluso disminuyan el LCoH, el dataset para el entrenamiento debería tener más datos con áreas de colectores y volumen de estanque de mayor valor, para que el modelo sea capaz de abarcar más configuraciones de búsqueda.

En el caso de las optimizaciones realizadas con los modelos de Machine Learning entrenados con el segundo dataset, lo primero que se puede notar son los errores que se obtienen en la validación de resultados. En primer lugar para Redes Neuronales, si se observan las tablas 4.24, 4.26 y 4.27 se puede notar que los errores son considerablemente más altos que los mencionados para Redes Neuronales con el primer dataset. En el caso del LCoH, para la optimización con un objetivo y multiobjetivo los errores no superan el 20 %, pero si se observan los errores para la energía de la caldera, son considerablemente altos, llegando a casi 71 %. La diferencia se puede deber a lo mencionado anteriormente. Quizás la energía de la caldera no tiene una incidencia muy grande dentro del cálculo del LCoH por ser lo suficientemente pequeña. Pero, al obtener un error tan alto para la energía de la caldera, los resultados previamente obtenidos no pueden ser considerados concluyentes. Esto se puede deber a los errores de testeo observados previamente en el entrenamiento del modelo, donde se mencionó que el modelo donde más tenía problemas era en la interpolación de esta energía. De hecho al observar los gráficos de valores test vs predichos, esta es la única que diverge del rango del 5 % de error. Y aunque sean pocos estos datos aislados, son los suficientes para que el modelo utilizado en la optimización no de resultados deseables. Lo que no ocurre para el primer dataset, donde la energía de la caldera entre los valores test y predichos nunca superan el rango del 5 % de error.

Lo mismo ocurre para el caso de Árboles de Decisión con el segundo dataset, donde si se observan las tablas 4.29, 4.31 y 4.32 los errores de validación son considerablemente altos, tanto para el LCoH como para la energía de la caldera. Lo que se debe a que el modelo en el entrenamiento tuvo resultados pobres como se mencionó anteriormente. Por estos resultados obtenidos, considerando que Árboles de Decisión con el segundo dataset era el modelo con peores resultados, era esperable que no dieran resultados coherentes dentro de la optimización. Y esto sumado al hecho de que, al igual que para Árboles con el primer dataset, se obtienen resultados muy similares para las configuraciones obtenidas y tampoco en el proceso de optimización se forma la curva de Pareto. Es decir, en este caso también ocurre que el modelo no interpola bien los resultados para el problema de optimización planteado, pero además los errores aumentan considerablemente debido a que en su entrenamiento el modelo tuvo errores altos.

En resumen, el buen comportamiento de las optimizaciones con los modelos de Machine Learning entrenados con el primer dataset se ve reflejado en los buenos resultados de entrenamientos de este, donde los errores de testeo eran considerablemente bajos. Mientras que en el caso del segundo dataset los errores comienzan a aumentar teniendo problemas al momento de interpolar la energía de la caldera en específico, lo que termina por causar que el modelo de optimización no de resultados concluyentes. Es decir, los errores de la optimización radican en el mal entrenamiento de los modelos de Machine Learning, que dependen de gran manera de la naturaleza del dataset que se utiliza para esto en el caso estudiado.

Capítulo 6

Conclusiones

En el trabajo realizado se presentaron diversas propuestas para el desarrollo de un modelo de optimización de una planta solar térmica integrada a procesos industriales. Bajo esta premisa y los resultados obtenidos, en primer lugar se puede concluir que se logró un buen planteamiento del problema de optimización y modelos capaces de resolver el problema. Se diseñaron modelos de Machine Learning con errores bajos, en particular para Redes Neuronales con el primer dataset, obteniendo entre 0.09 y 0.12 % de error. Se logró una buena comparación entre Árboles de Decisión y Redes Neuronales, donde se concluye que este último tiene mejor rendimiento para el problema particular planteado. También la comparación de diferentes técnicas de preprocesamiento, donde las normalizaciones obtuvieron mejores resultados, en particular Standard Scaler.

Con respecto a los modelos de Machine Learning, se puede mencionar que en el desarrollo del trabajo se logró implementar modelos con alta capacidad de predicción de las energías asociadas a la planta solar y bajo error porcentual. Con respecto a esto, lo más importante que se pudo notar es la elección del preprocesamiento de datos antes del entrenamiento del modelo, y se vuelve a mencionar que este paso es sumamente importante y necesario para la obtención de buenos resultados. Comparando los dos modelos de Machine Learning utilizados, se puede concluir que para la obtención de los outputs planteados en este trabajo el modelo con mejores resultados corresponde a Redes Neuronales. Esto se vio tanto para los entrenados con el primer y segundo dataset, donde se obtuvieron errores de testeo más bajos que Árboles de Decisión, teniendo una diferencia de 0.05 % para el primer dataset y 3.11 % para el segundo dataset. Es decir, el modelo de Redes Neuronales para este caso predice e interpola los valores de mejor manera que Árboles de Decisión.

Los modelos de optimización, tanto para un objetivo y multiobjetivo logran minimizar el LCoH y la energía de la caldera con sus respectivas restricciones. Evaluando los resultados obtenidos, se pudo notar que a pesar de que todos los modelos de optimización tenían la misma arquitectura pero con los distintos modelos de Machine Learning, no se obtuvieron buenos resultados en todos los casos. En particular, tanto para Redes Neuronales y Árboles de Decisión entrenados con el segundo dataset se obtuvieron resultados con un alto error porcentual en algunos casos, los mayores siendo 70.92 % y 157.27 % respectivamente. En

comparación a Redes Neuronales con el primer dataset que no superan el 2 % de error con respecto a los resultados nuevamente simulados. Por esto, se concluye que los modelos de Machine Learning utilizados deben tener errores muy bajos para obtener buenos resultados en la optimización. Es decir, los datos predichos por el modelo no debe superar un error del 5 % con respecto al dato real, para evitar obtener datos aislados en la optimización que alteren los resultados, como ocurre con Redes Neuronales entrenadas con el segundo dataset comparándolo con el caso del primer dataset.

Por lo mencionado anteriormente, se puede concluir que el entrenamiento de las Redes Neuronales para ser utilizado en el modelo de optimización es mejor para un set de datos más homomogéneo con valores equiespaciados, que uno con datos al azar. Como se mencionó en el análisis las Redes Neuronales son capaces de interpolar de buena manera datos que no se encontraban en el dataset, mientras que quizás tener datos tan variados en el entrenamiento dificulta el aprendizaje de la Red, aumentando de gran manera el error de predicción de la energía de la caldera en comparación a las otras.

Con lo mencionado, se puede concluir que se logró desarrollar y aplicar un modelo de optimización basado en algoritmos de Machine Learning que sean capaces de entregar resultados coherentes y con bajo error, el cual sería el modelo de optimización multiobjetivo con el modelo de Redes Neuronales entrenado con el primer dataset entregado con Standard Scaler.

En resumen, para obtener resultados coherentes y con bajo error con el modelo de optimización, lo más importante corresponde a utilizar un modelo de Redes Neuronales bien entrenadas y con un error lo suficientemente bajo, esto para evitar obtener datos aislados en la optimización que terminen por alterar los resultados. Y para obtener un modelo de Redes Neuronales con bajo error, es importante el preprocesamiento de datos que se utilice, así como también la naturaleza de los datos. También se concluye que para este caso mencionado el modelo a priori puede ser utilizado en la práctica para optimizar plantas solares térmicas, teniendo en consideración las diferencias entre simulaciones de estas y resultados reales.

Trabajo Propuesto

Queda como trabajo propuesto aplicar el modelo de optimización multiobjetivo con Redes Neuronales entrenadas con un dataset de la misma naturaleza que el primero utilizado, pero más amplio. Es decir, con valores mayores de áreas de colectores, volúmenes, etc. para evitar la saturación observada del modelo.

Además, podría ser interesante ver cómo se comporta el modelo multiobjetivo minimizando el LCoH, pero con otro objetivo que influya más en este que la minimización de la caldera. Esto para observar de mejor manera el comportamiento del LCoH. Además, para la industria en la práctica la minimización de la energía de la caldera no sea lo más deseable, si no que también se quiera minimizar otro indicador de costos o maximizar la eficiencia

de la planta solar por ejemplo. El modelo planteado queda abierto a que se modifiquen las funciones objetivos para obtener resultados que faciliten de mejor manera los proyectos a minimizar costos y maximizar beneficios.

Por ultimo, se podría aplicar el modelo, pero entrenando las Redes Neuronales con más validaciones cruzadas, o ampliando el conjunto de búsqueda de los hiperparámetros y cantidad de modelos entrenados en el tuner. Para observar si esto se traduce en la disminución de los errores de los modelos de Machine Learning para mejores resultados en la optimización. Tomando en consideración que esto aumentaría el tiempo de ejecución del algoritmo, y que también se necesitarían más recursos computacionales que los utilizados en este trabajo.

En general, para una aplicación de un modelo de planta solar térmica en particular, el trabajo realizado tiene buena capacidad de predicción y bajo errores de optimización, por lo que puede ser una propuesta eficaz para problemas como el estudiado. Pero para finalizar, el trabajo también podría ser ampliado a una optimización más general. Por ejemplo, optimizar la locación de la planta solar utilizando como inputs datos meteorológicos del país o que sea adaptable a un proceso industrial de preferencia adaptando la demanda dentro del algoritmo de Machine Learning.

Bibliografía

[1] Ministerio de Energía, "Balance nacional de energía año 2020," 2021, https://energia.gob.cl/pelp/balance-nacional-de-energia. 1

[2] Ministerio de Energía, "Explorador solar," 2017, https://solar.minenergia.cl/inicio. 1

[3] Solar Heating And Cooling Programme, "Solar heat integrations in industrial processes," 2020, https://task49.iea-shc.org/Data/Sites/1/publications/IEA-SHC-Technology-Position-Paper--Solar-Heat-Integrations-Industrial-Processes--May2020.pdf. 1

[4] Cortés, F., Ibarra, M., Moser, F., Muñoz, I., Crespo, A., y Murray, C., "Techno-economical evaluation of parabolic trough collectors systems for steam processes in the chilean industry," AIP Conference Proceedings, vol. 2033, no. 1, p. 150003, 2018, doi:10.1063/1.5067156. 1, 3

[5] The United Nations Environment Programme, "Solar heat for industrial process," 2017, http://solarheateurope.eu/project/solar-heat-industrial-process/. 3

[6] Muster-Slawitsch, B., "Integrating solar heat into industrial processes (ship)," Solar Heating and Cooling Programme, 2016, https://wvvvv.iea-shc.org/publications-category?CategoryID=396. 4

[7] Cottret, N. y Menichetti, E., "Technical study report on solar heat for industrial processes (ship) state of the art in the mediterranean region," Observatoire Méditerranéen de l'Energie: Nanterre, France, 2010, https://www.solarthermalworld.org/sites/default/files/story/2015-10-14/solar_heat_for_industrital_process_technical_report._state_of_the_art_in_the_mediterranean_region.pdf. 4

[8] Revista Electricidad, "Planta termosolar tiene un avance de 70 % en división gabriela mistral," 2013, https://www.revistaei.cl/2013/06/12/planta-termosolar-tiene-un-avance-de-70-en-division-gabriela-mistral/#. 4

[9] Delgado, J. I., "Estudio técnico y económico para la instalación de energía solar fotovoltaica residencial conectado a la red de distribución. [memoria de titulación para optar al título de ingeniero civil mecánico]," 2021, https://repositorio.usm.cl/bitstream/handle/11673/50617/3560902038071UTFSM.pdf?sequence=1&isAllowed=y. 4

[10] SOLARGIS, "Mapas de recursos solares de chile," 2017, https://solargis.com/es/maps-and-gis-data/download/chile. ix, 5

[11] Kalogirou, S., "Artificial neural networks and genetic algorithms for the optimisation of solar thermal systems," Artificial Intelligence in Energy and Renewable Energy Systems,

2006, https://ktisis.cut.ac.cy/bitstream/10488/824/2/C82-Inv\%20ST7.pdf. 6, 13

[12] Helin, R., Indahl, U. G., Tomic, O., y Liland, K. H., "On the possible benefits of deep learning for spectral preprocessing," Journal of Chemometrics, vol. 36, no. 2, p. e3374, 2022, https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/full/10.1002/cem.3374. 6

[13] Herrera, F., "Big data: Preprocesamiento y calidad de datos," novática, vol. 237, p. 17, 2016, https://sci2s.ugr.es/sites/default/files/ficherosPublicaciones/2133_Nv237-Digital-sramirez.pdf. 6, 7

[14] Scikit-Learn, "Api scikit-learn version 1.2.0," 2022, https://scikit-learn.org/stable/index.html. 6

[15] Abdi, H. y Williams, L. J., "Principal component analysis," WIREs Computational Statistics, vol. 2, no. 4, pp. 433–459, 2010, doi:https://doi.org/10.1002/wics.101. 7

[16] Liu, Z., Liu, K., y Zhang, Z., "Predictive power of machine learning for optimizing solar water heater performance: The potential application of high-throughput screening," International Journal of Photoenergy, vol. 2017, pp. 1–10, 2017, doi:10.1155/2017/4194251. ix, 7

[17] Xu, M., Watanachaturaporn, P., Varshney, P. K., y Arora, M. K., "Decision tree regression for soft classification of remote sensing data," Remote Sensing of Environment, vol. 97, no. 3, pp. 322–336, 2005. 8

[18] Chiu, M.-H., Yu, Y.-R., Liaw, H., y Hao, L., "The use of facial micro-expression state and tree-forest model for predicting conceptual-conflict based conceptual change," 2016. ix, 8

[19] Forrest, S., "Genetic algorithms," ACM Computing Surveys (CSUR), vol. 28, no. 1, pp. 77–80, 1996, https://dl.acm.org/doi/pdf/10.1145/234313.234350. 9

[20] Kramer, O., "Genetic algorithms," en Genetic algorithm essentials, pp. 11–19, Springer, 2017, doi:10.1007/978-3-319-52156-5_2. 9

[21] Gad, A., "Pygad: An intuitive genetic algorithm python library," 2021, https://www.researchgate.net/publication/352373466_PyGAD_An_Intuitive_Genetic_Algorithm_Python_Library. ix, 9, 10, 23

[22] Deb, K., Pratap, A., Agarwal, S., y Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE transactions on evolutionary computation, vol. 6, no. 2, pp. 182–197, 2002. ix, 10, 11

[23] Weichert, D., Link, P., Stoll, A., Rüping, S., Ihlenfeldt, S., y Wrobel, S., "A review of machine learning for the optimization of production processes," The International Journal of Advanced Manufacturing Technology, vol. 104, no. 5, pp. 1889–1902, 2019. 11, 12

[24] Lee, J., Hong, S., Cho, H., Lyu, B., Kim, M., Kim, J., y Moon, I., "Machine learning-based energy optimization for on-site smr hydrogen production," Energy Conversion and Management, vol. 244, p. 114438, 2021. ix, 11, 12

[25] Magnier, L. y Haghighat, F., "Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network," Building and Environment, vol. 45, no. 3, pp. 739–746, 2010. 12

[26] Fadlallah, S. O., Anderson, T. N., y Nates, R. J., "Artificial neural network–particle swarm optimization (ann-pso) approach for behaviour prediction and structural optimization of lightweight sandwich composite heliostats," Arabian Journal for Science and Engineering, vol. 46, no. 12, pp. 12721–12742, 2021. 13

[27] Dallapiccola, M., Trentin, F., Barchi, G., Dipasquale, C., Fedrizzi, R., y Moser, D., "Machine learning driven optimization of a hybrid electrical and thermal system," en Proceedings of the 37th European Photovoltaic Solar Energy Conference and Exhibition, Lisboa, Portugal, pp. 7–11, 2020. ix, 13

[28] Starke, A. R., Cardemil, J. M., Escobar, R., y Colle, S., "Multi-objective optimization of hybrid csp+ pv system using genetic algorithm," Energy, vol. 147, pp. 490–503, 2018. 13

[29] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., y Varoquaux, G., "API design for machine learning software: experiences from the scikit-learn project," en ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122, 2013. 17

[30] Chollet, F., "Keras." https://github.com/fchollet/keras, 2019. 19

[31] O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., *et al.*, "Keras Tuner." https://github.com/keras-team/keras-tuner, 2019. 19

[32] Mardini, D. M., "Identificación de daño en paneles compuestos utilizando redes neuronales convolucionales y los modos de vibración. [memoria para optar al título de ingeniero civil mecánico]," 2020, https://repositorio.uchile.cl/handle/2250/178521. 21

[33] Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., y Prasath, V. S., "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," Information, vol. 10, no. 12, p. 390, 2019. 26

[34] Blank, J. y Deb, K., "pymoo: Multi-objective optimization in python," IEEE Access, vol. 8, pp. 89497–89509, 2020. 26

[35] Hamdan, M., "The distribution index in polynomial mutation for evolutionary multiobjective optimisation algorithms: An experimental study," en International Conference on Electronics Computer Technology (IEEE, Kanyakumari, India, 2012), 2012. ix, 28

Anexos

Anexo A. Gráficos de Pérdida de Redes Neuronales con primer dataset

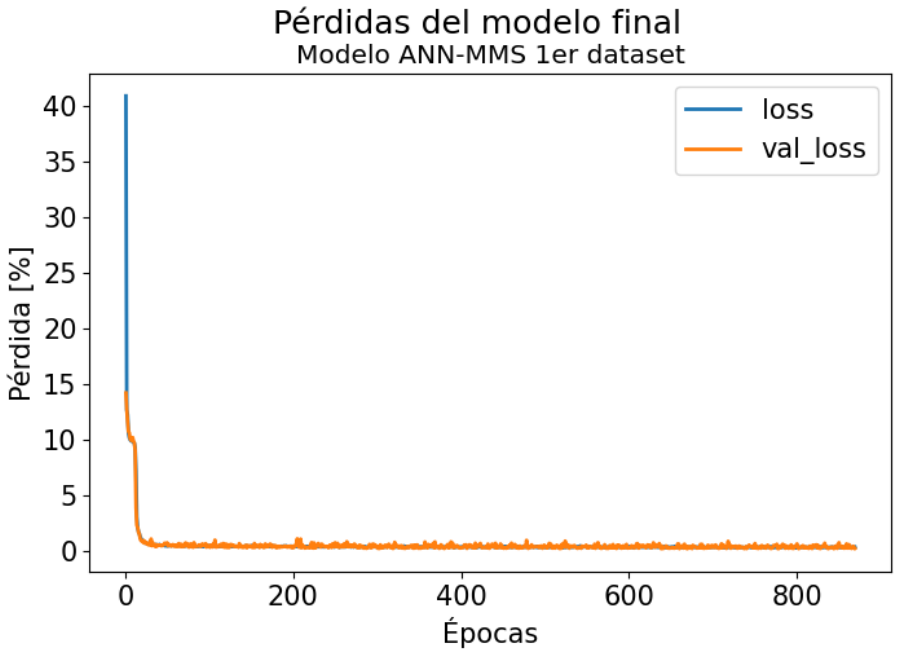


Figura A.1: Gráfico de pérdidas modelo ANN-MMS con primer dataset

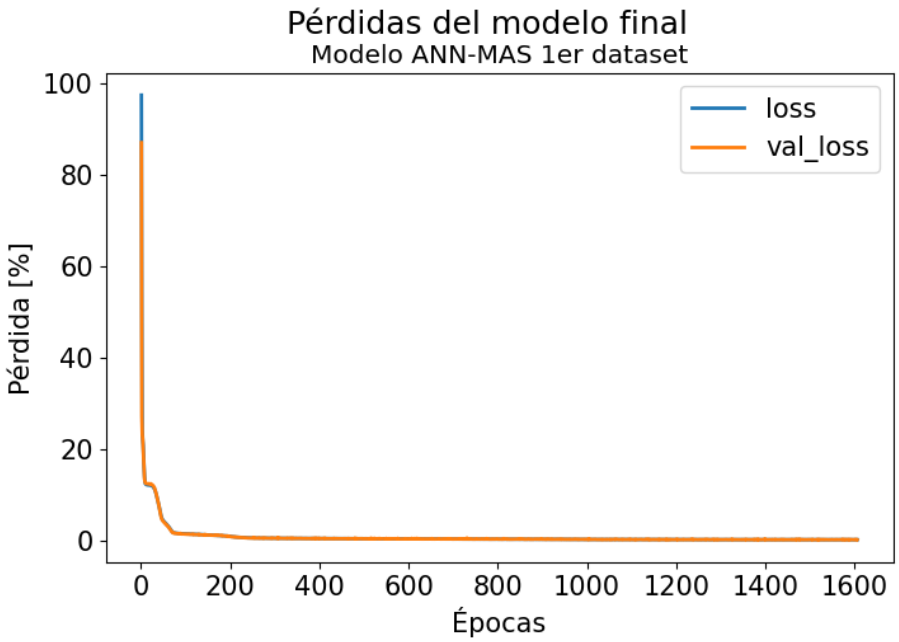


Figura A.2: Gráfico de pérdidas modelo ANN-MAS con primer dataset

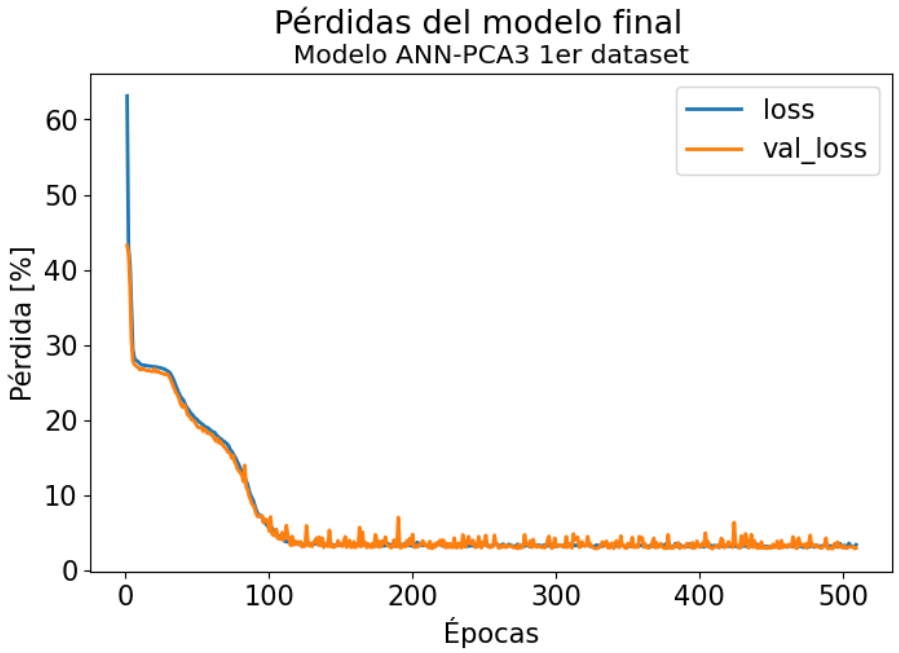


Figura A.3: Gráfico de pérdidas modelo ANN-PCA3 con primer dataset

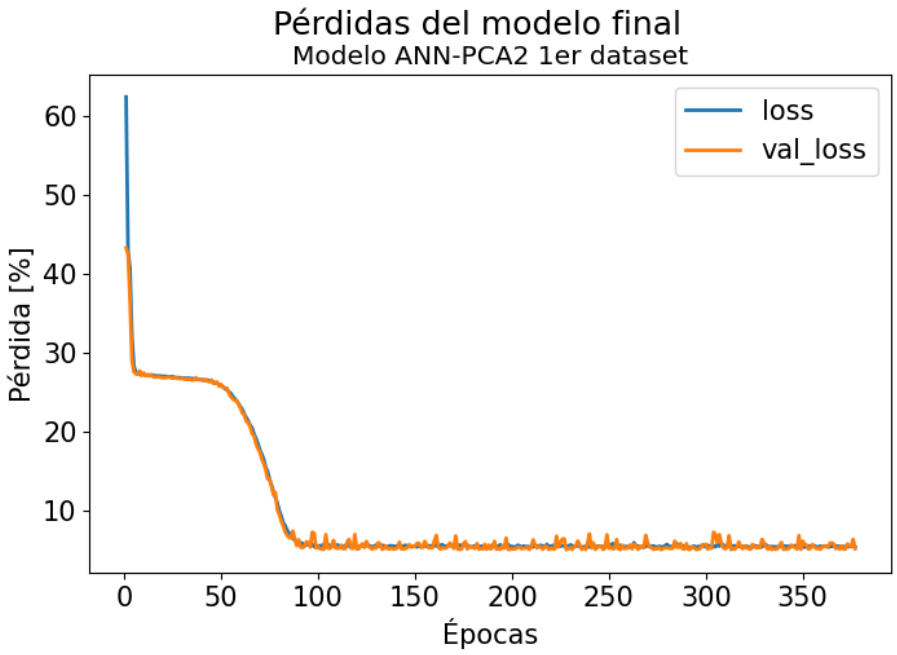


Figura A.4: Gráfico de pérdidas modelo ANN-PCA2 con primer dataset

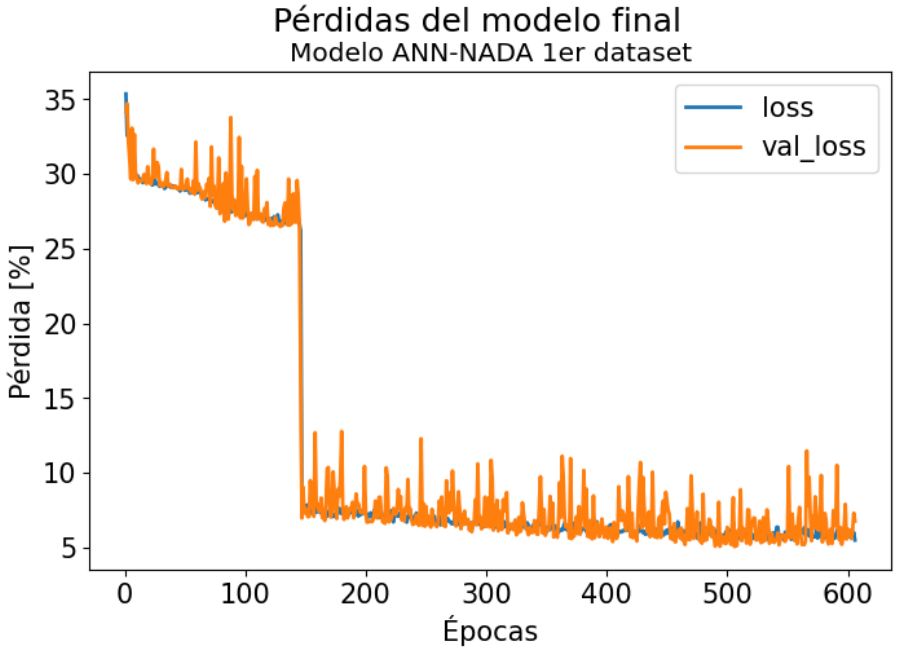


Figura A.5: Gráfico de pérdidas modelo ANN-NADA con primer dataset

Anexo B. Learning Curve de Árboles de Decisión con primer dataset

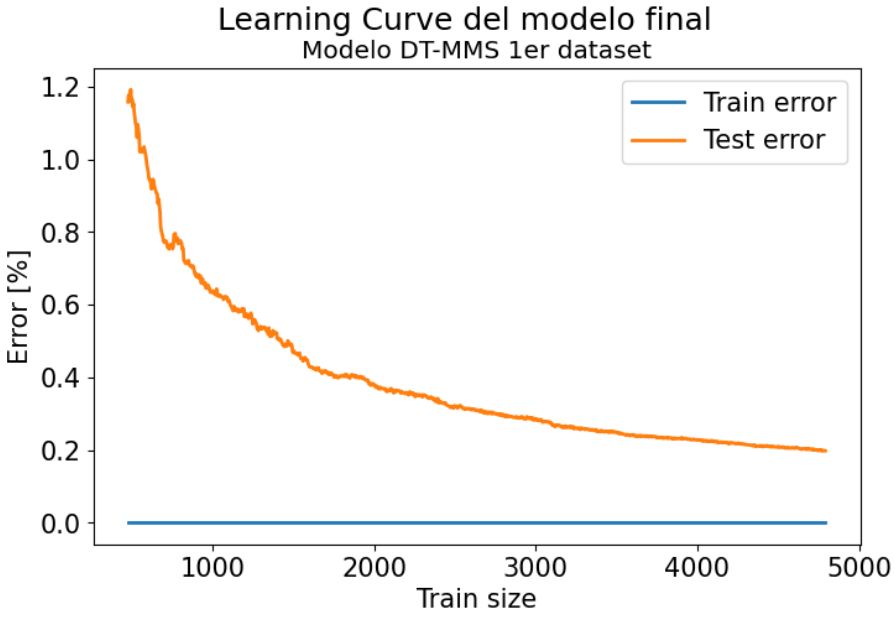


Figura B.1: Learning Curve modelo DT-MMS con primer dataset

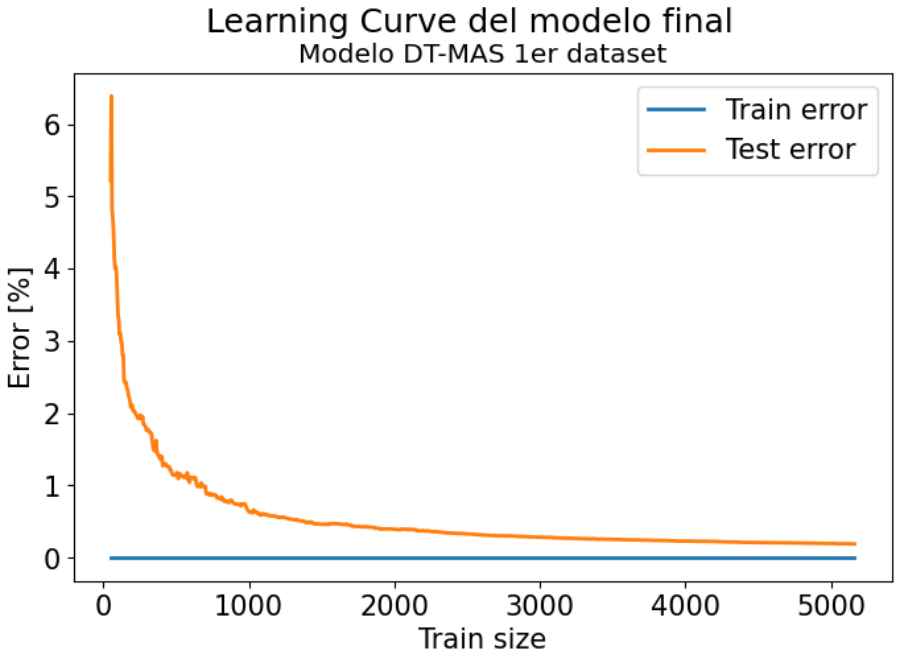


Figura B.2: Learning Curve modelo DT-MAS con primer dataset

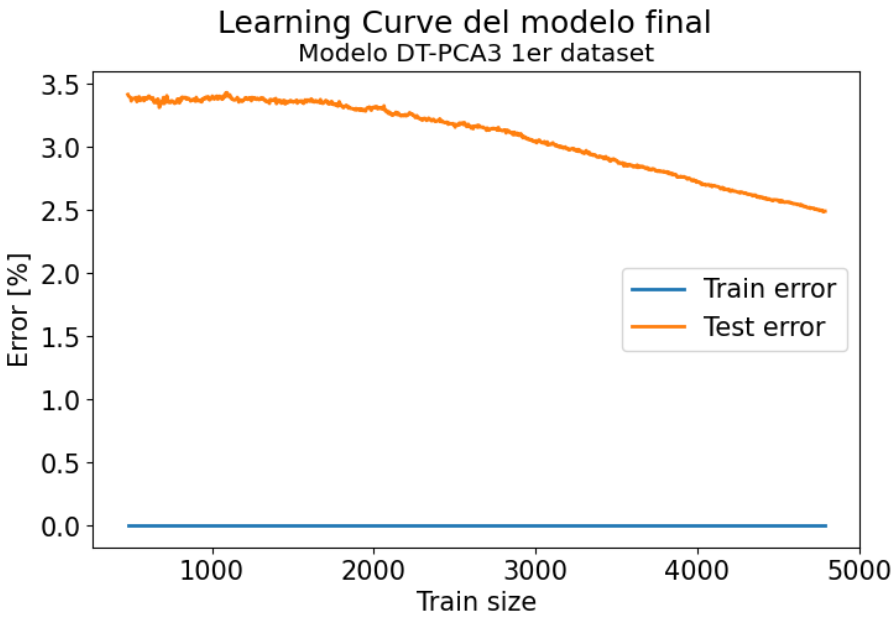


Figura B.3: Learning Curve modelo DT-PCA3 con primer dataset

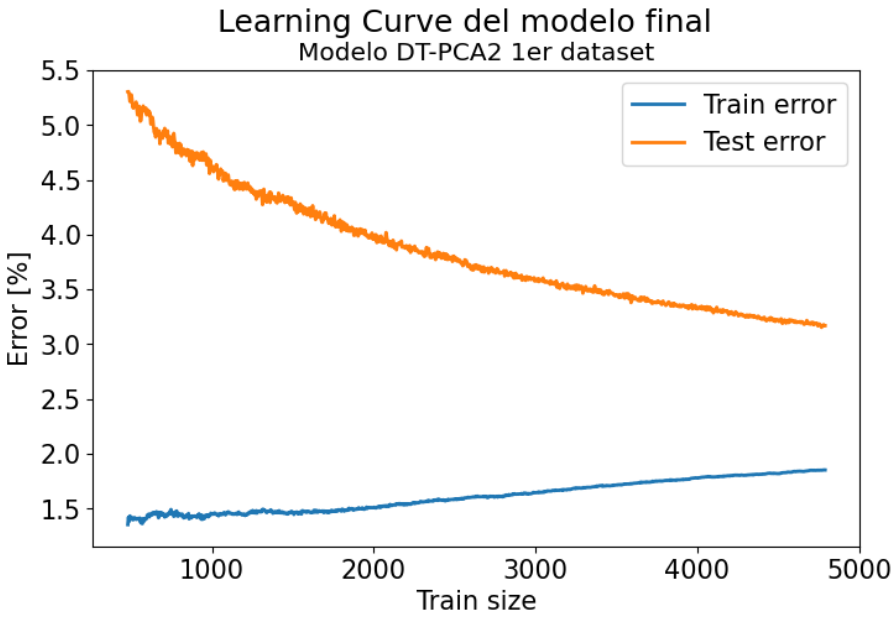


Figura B.4: Learning Curve modelo DT-PCA2 con primer dataset

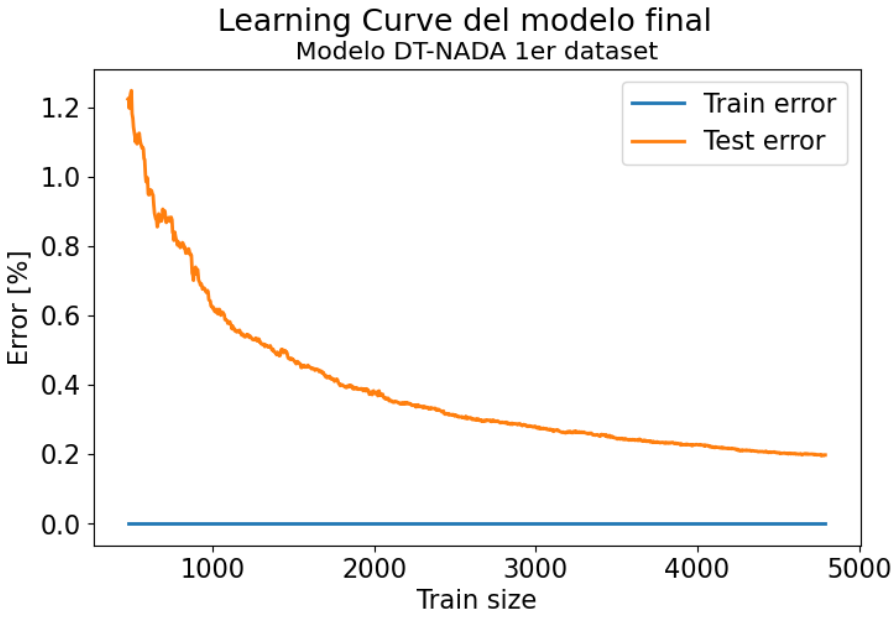


Figura B.5: Learning Curve modelo DT-NADA con primer dataset

Anexo C. Resultados para Redes Neuronales con primer dataset

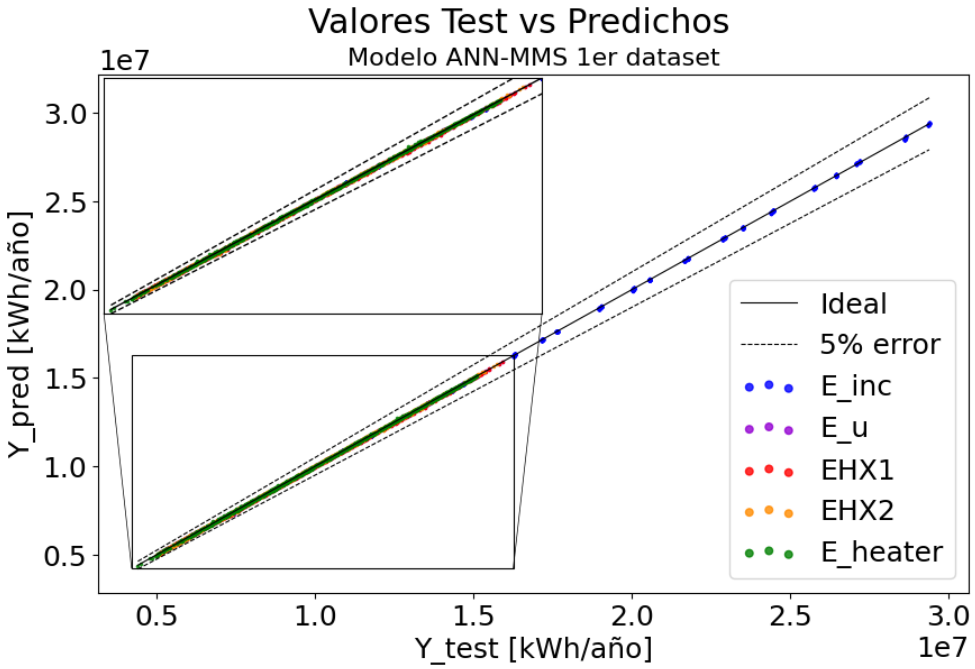


Figura C.1: Valores test vs predichos modelo ANN-MMS con primer dataset

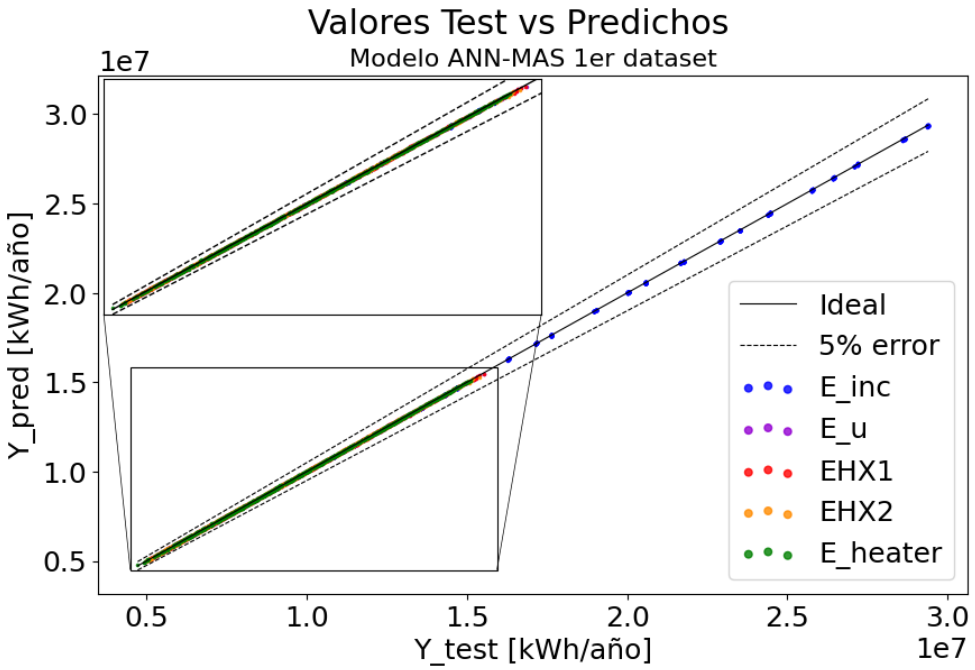


Figura C.2: Valores test vs predichos modelo ANN-MAS con primer dataset

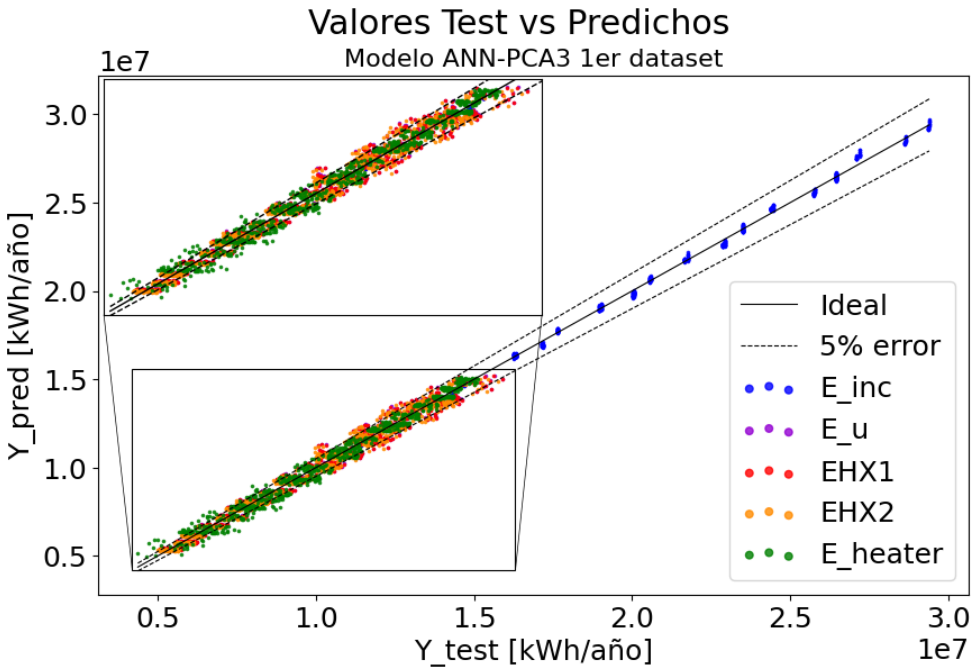


Figura C.3: Valores test vs predichos modelo ANN-PCA3 con primer dataset

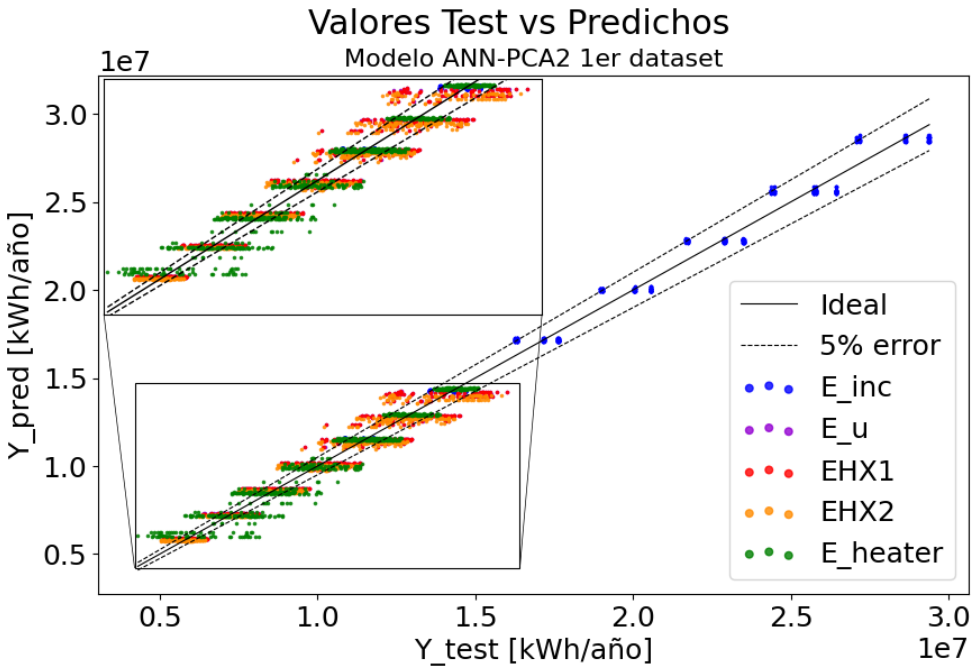


Figura C.4: Valores test vs predichos modelo ANN-PCA2 con primer dataset

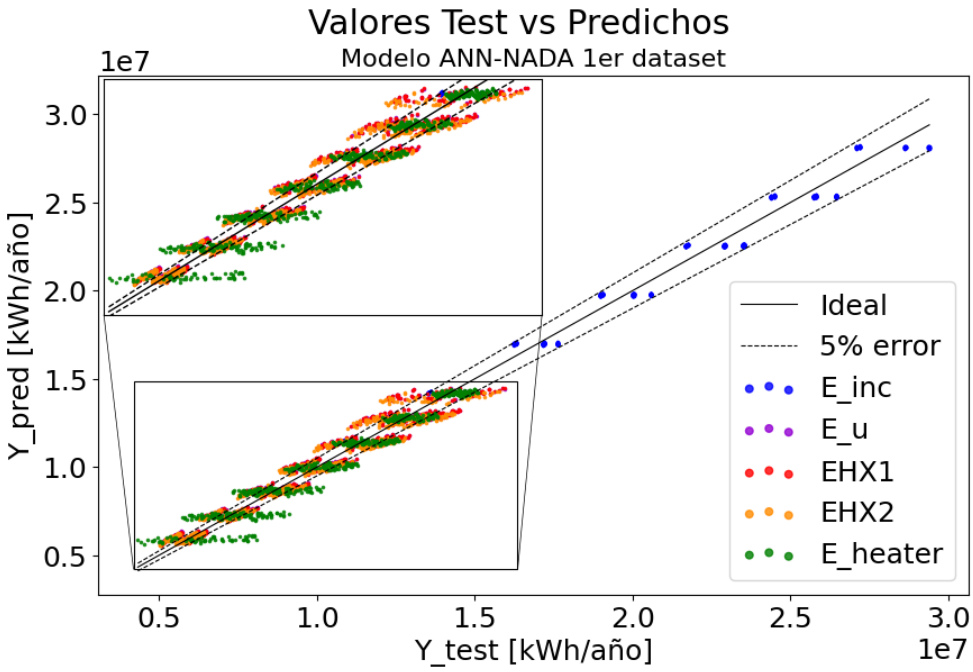


Figura C.5: Valores test vs predichos modelo ANN-NADA con primer dataset

Anexo D. Resultados para Árboles de Decisión con primer dataset

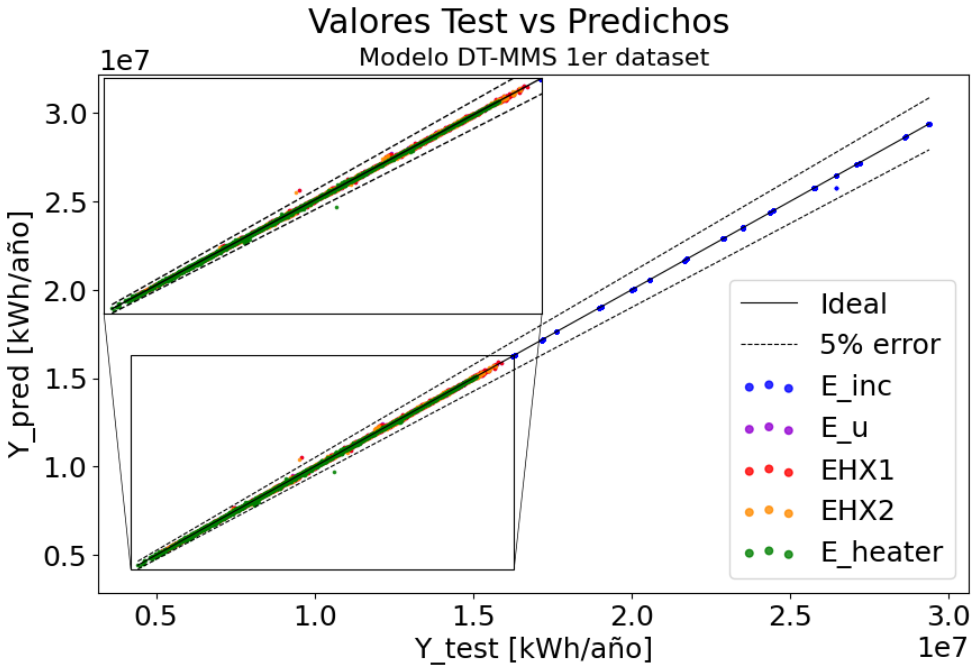


Figura D.1: Valores test vs predichos modelo DT-MMS con primer dataset

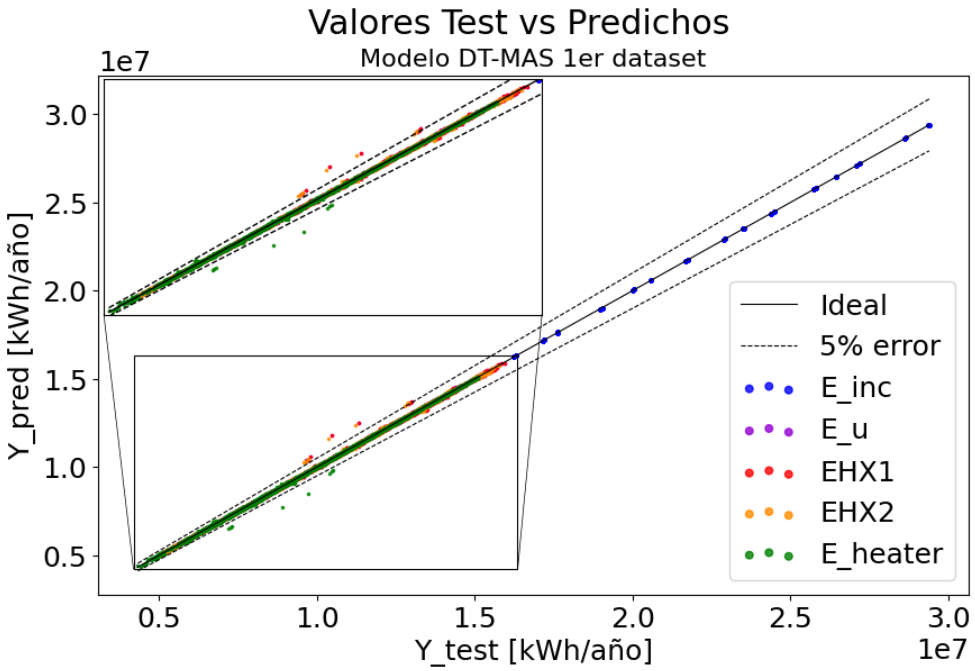


Figura D.2: Valores test vs predichos modelo DT-MAS con primer dataset

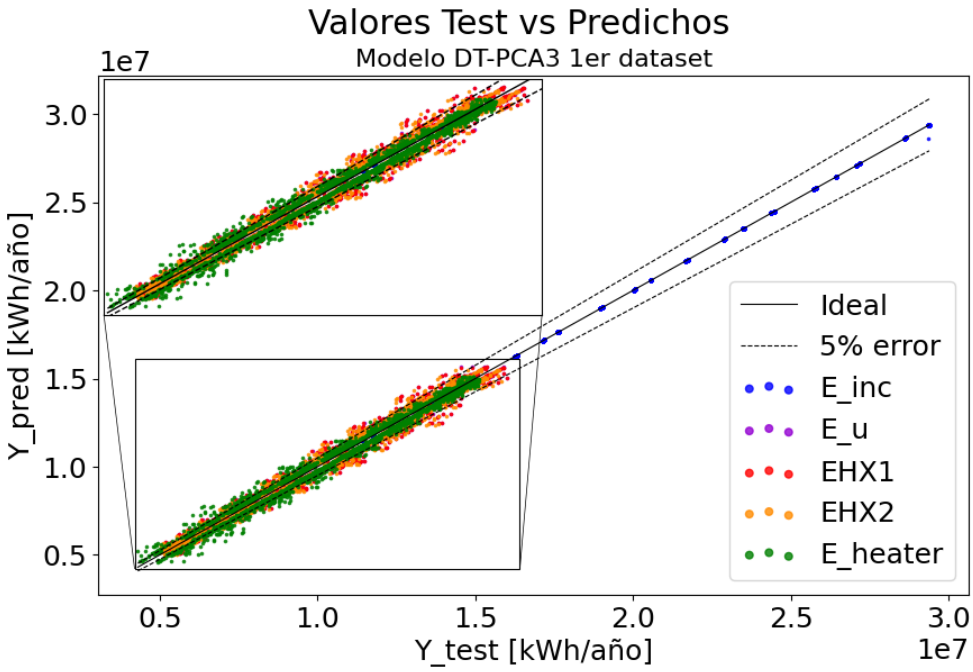


Figura D.3: Valores test vs predichos modelo DT-PCA3 con primer dataset

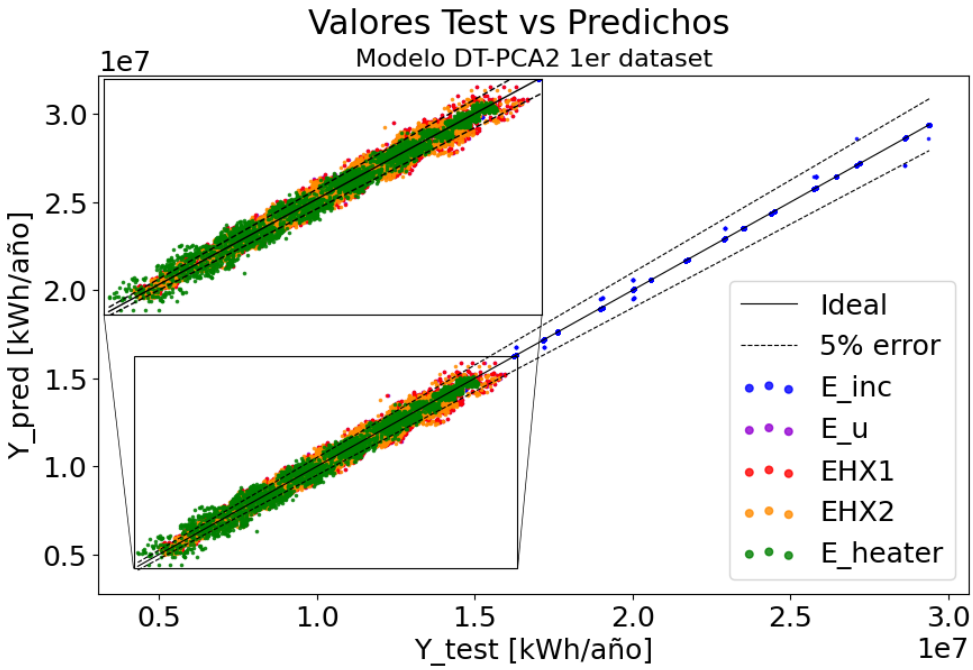


Figura D.4: Valores test vs predichos modelo DT-PCA2 con primer dataset

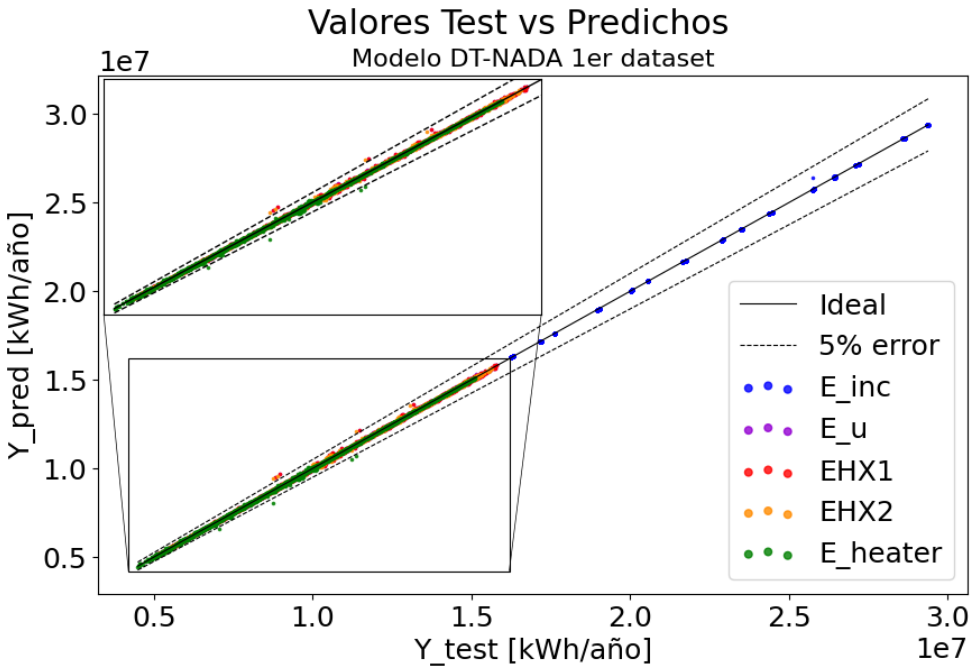


Figura D.5: Valores test vs predichos modelo DT-NADA con primer dataset

Anexo E. Gráficos de Pérdida de Redes Neuronales con segundo dataset

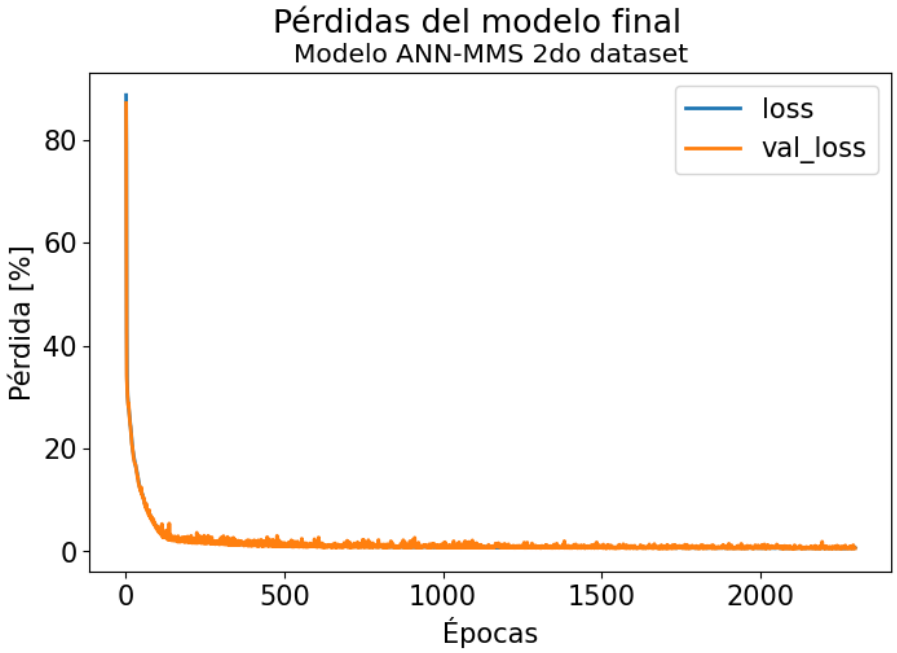


Figura E.1: Gráfico de pérdidas modelo ANN-MMS con segundo dataset

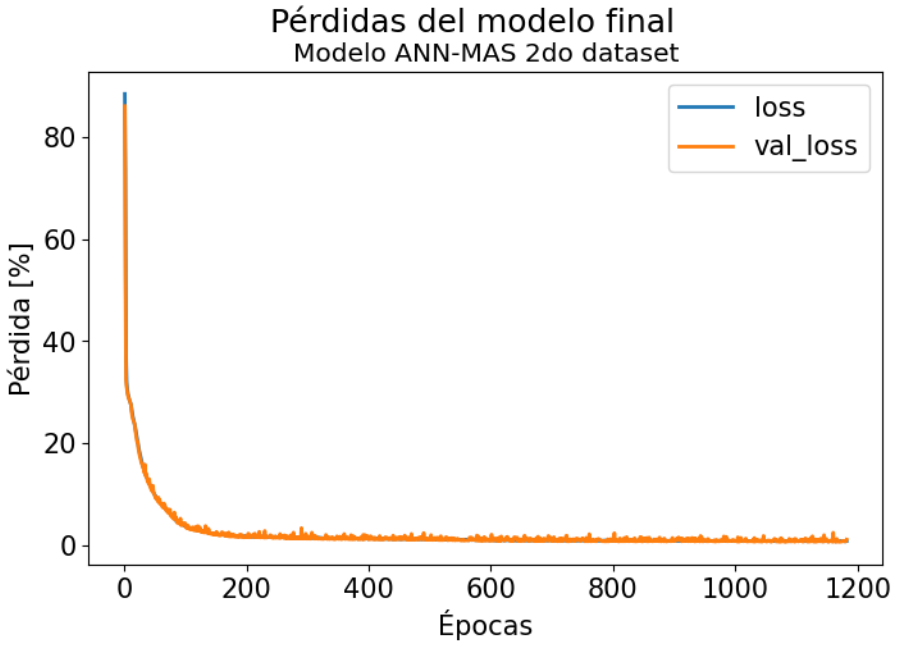


Figura E.2: Gráfico de pérdidas modelo ANN-MAS con segundo dataset

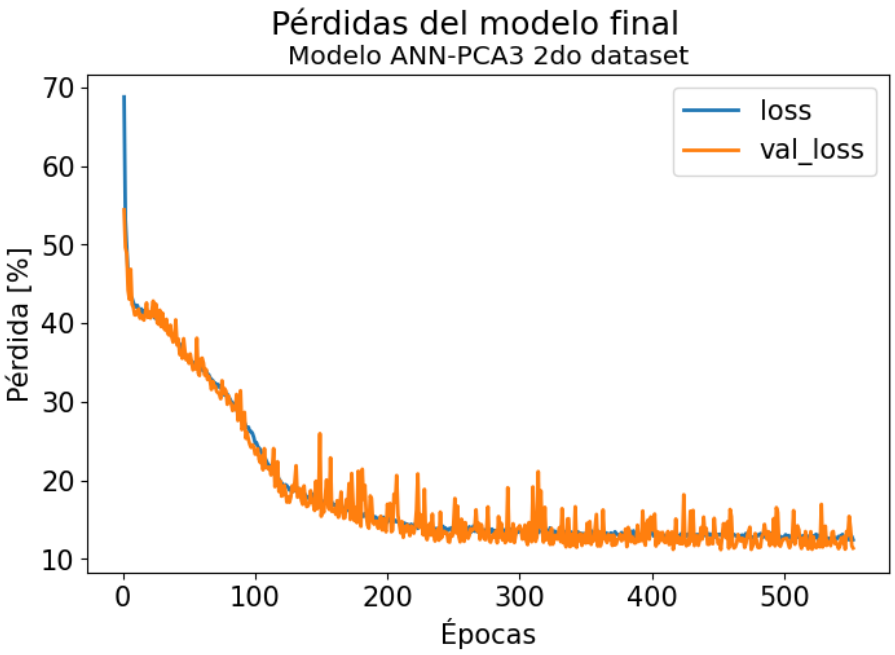


Figura E.3: Gráfico de pérdidas modelo ANN-PCA3 con segundo dataset

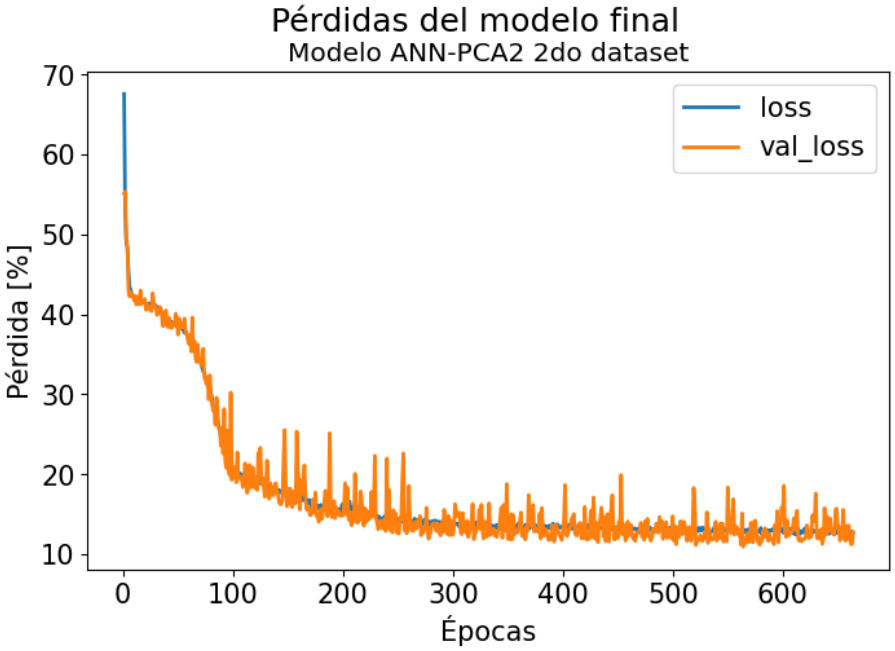


Figura E.4: Gráfico de pérdidas modelo ANN-PCA2 con segundo dataset

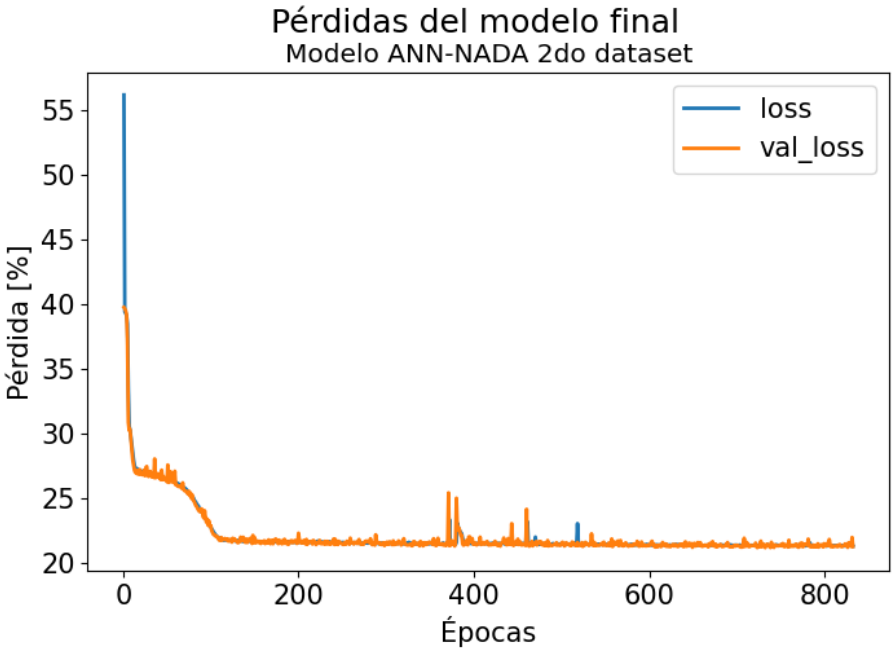


Figura E.5: Gráfico de pérdidas modelo ANN-NADA con segundo dataset

Anexo F. Learning Curve de Árboles de Decisión con segundo dataset

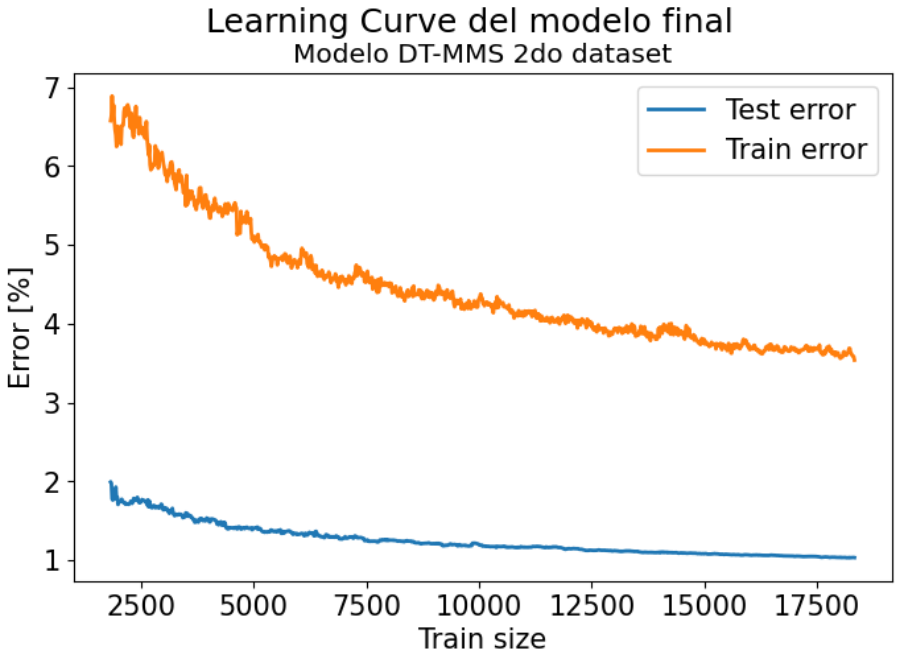


Figura F.1: Learning Curve modelo DT-MMS con segundo dataset

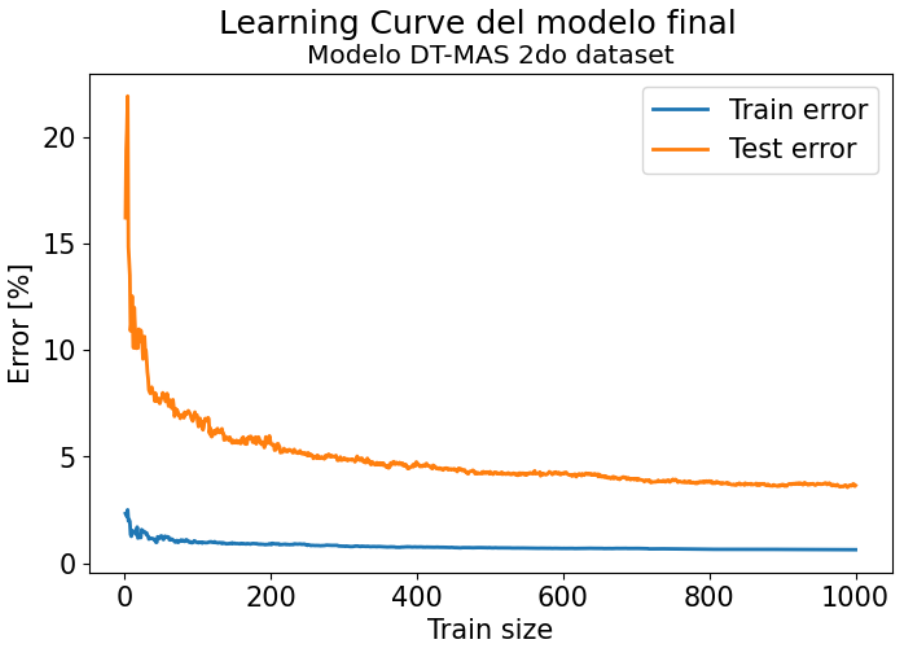


Figura F.2: Learning Curve modelo DT-MAS con segundo dataset

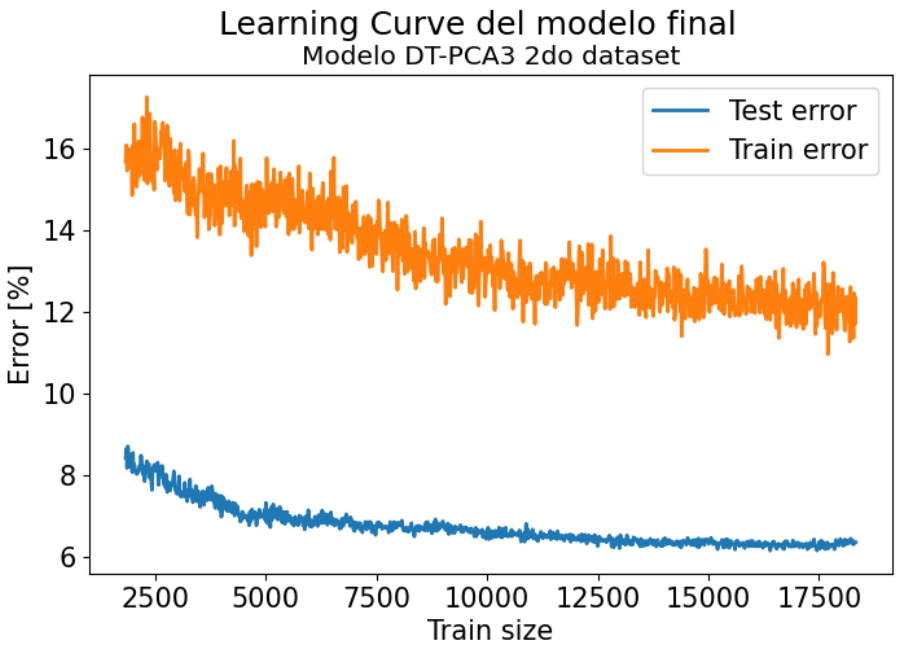


Figura F.3: Learning Curve modelo DT-PCA3 con segundo dataset

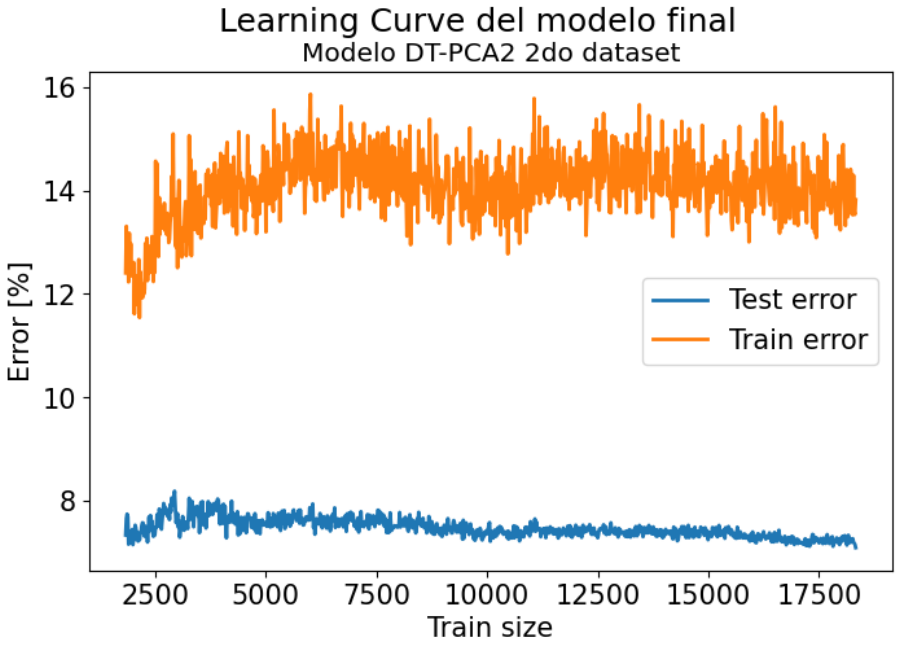


Figura F.4: Learning Curve modelo DT-PCA2 con segundo dataset

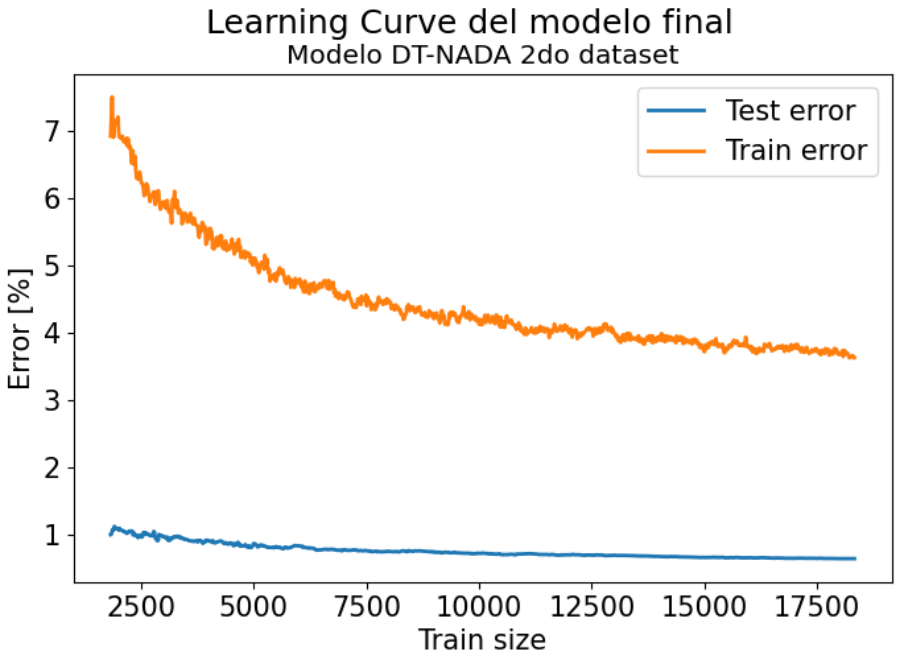


Figura F.5: Learning Curve modelo DT-NADA con segundo dataset

Anexo G. Resultados para Redes Neuronales con segundo dataset

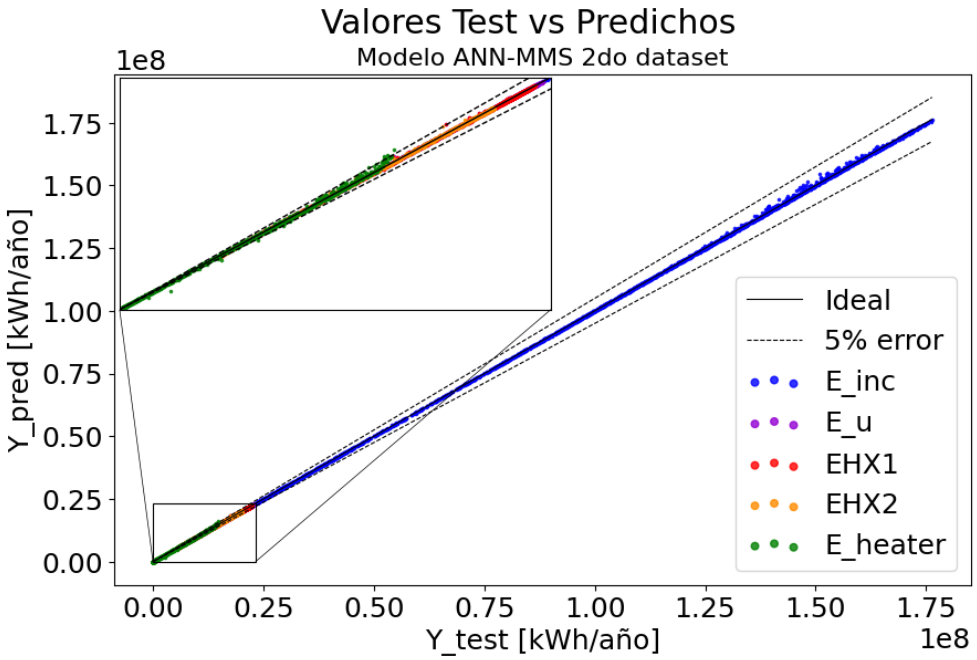


Figura G.1: Valores test vs predichos modelo ANN-MMS con segundo dataset

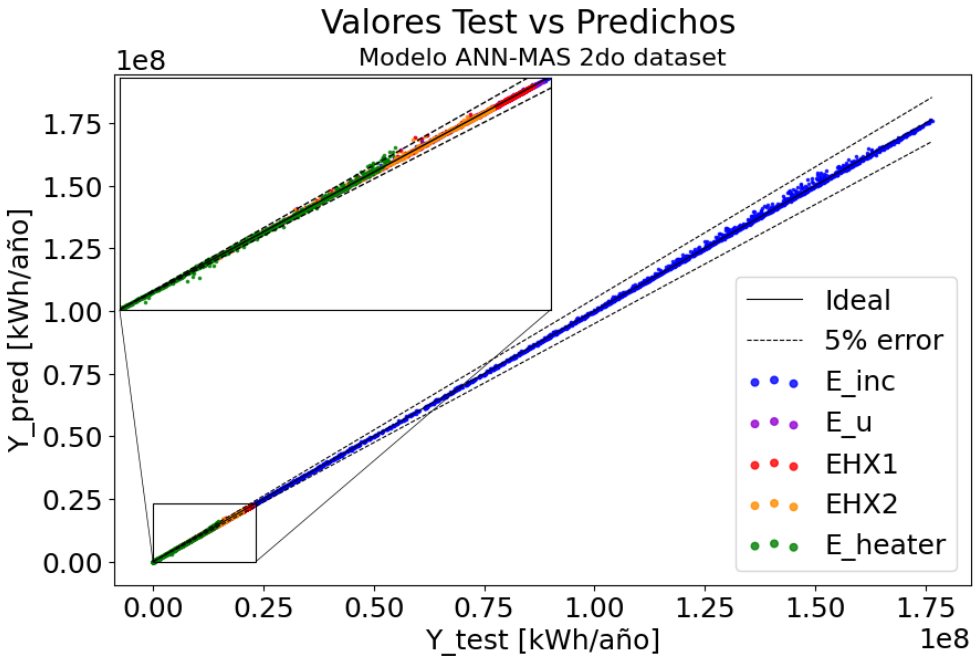


Figura G.2: Valores test vs predichos modelo ANN-MAS con segundo dataset

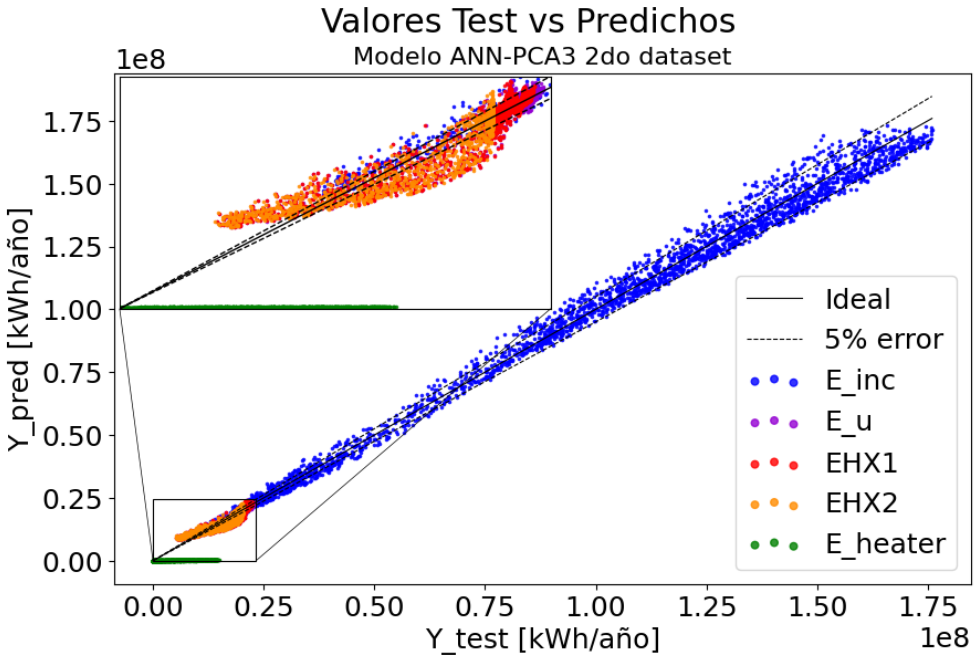


Figura G.3: Valores test vs predichos modelo ANN-PCA3 con segundo dataset

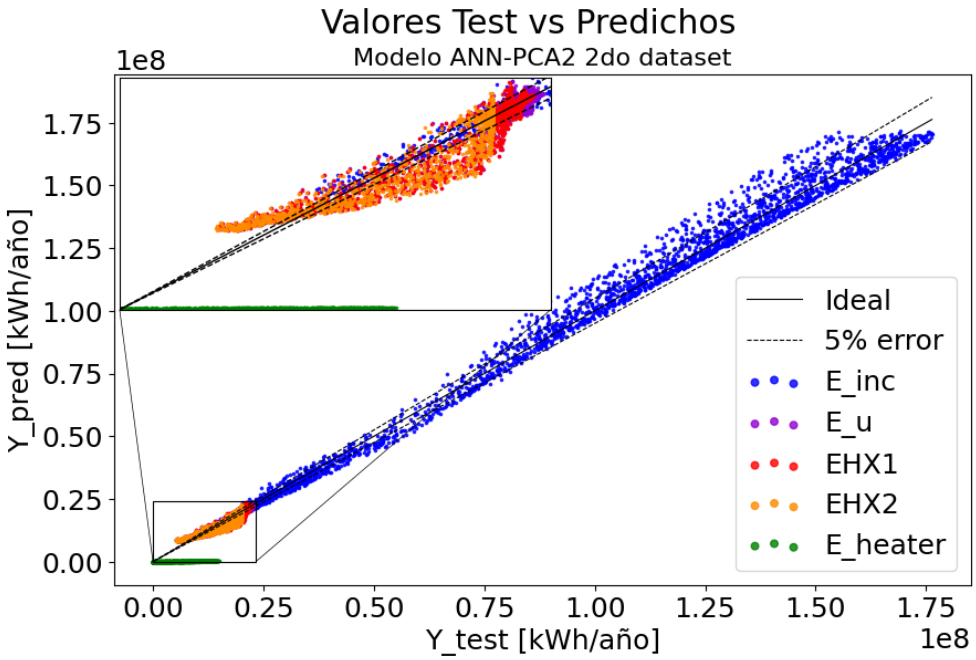


Figura G.4: Valores test vs predichos modelo ANN-PCA2 con segundo dataset

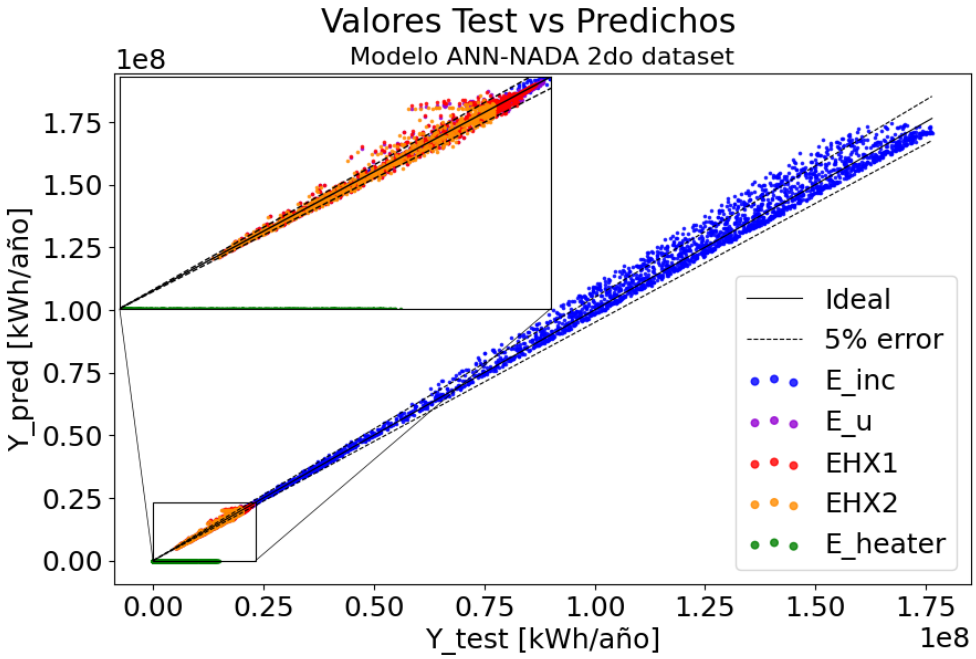


Figura G.5: Valores test vs predichos modelo ANN-NADA con segundo dataset

Anexo H. Resultados para Árboles de Decisión con segundo dataset

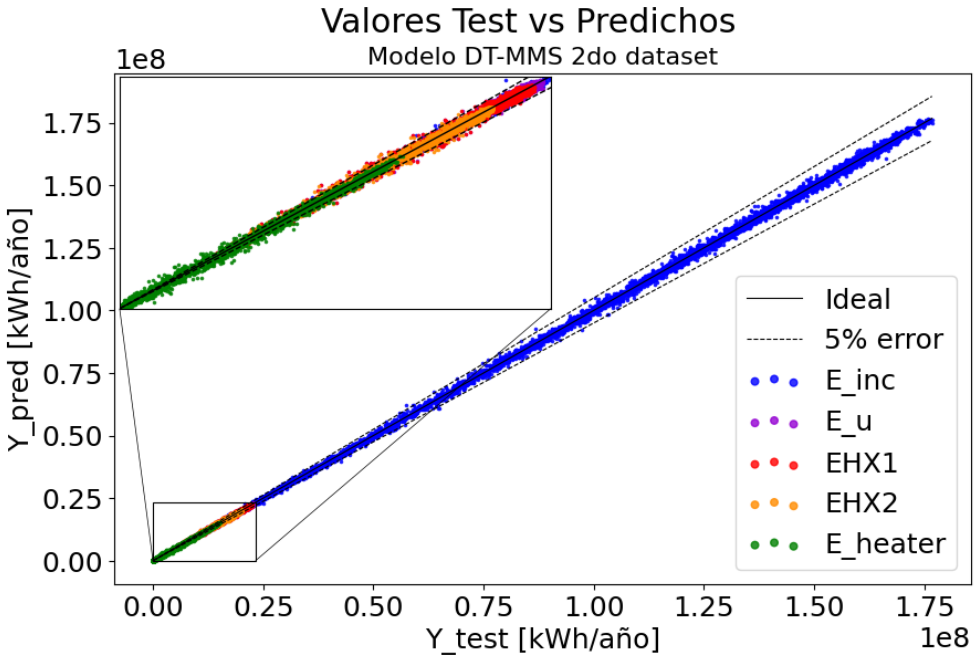


Figura H.1: Valores test vs predichos modelo DT-MMS con segundo dataset

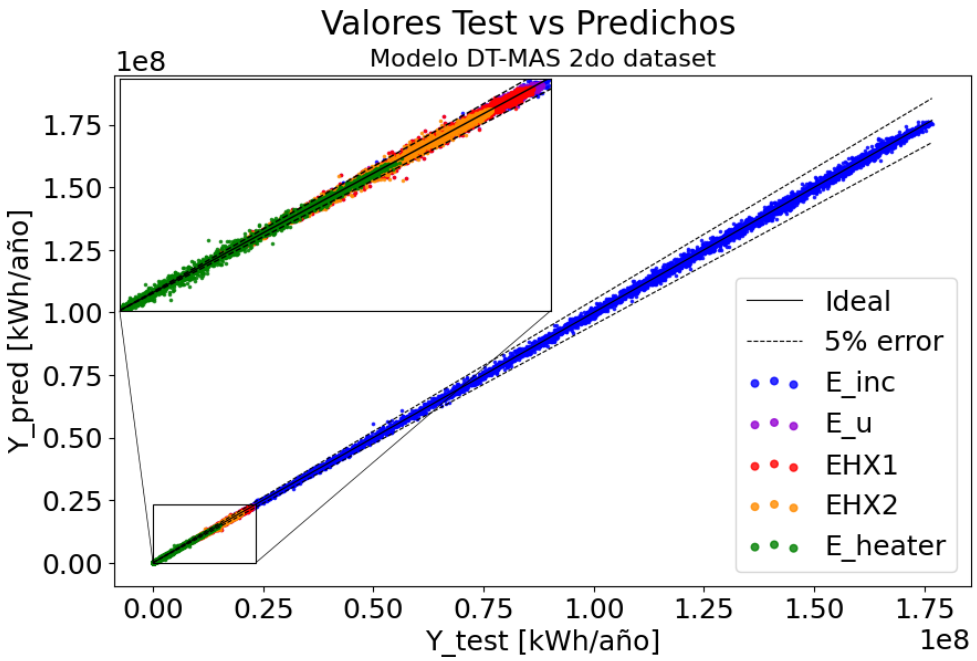


Figura H.2: Valores test vs predichos modelo DT-MAS con segundo dataset

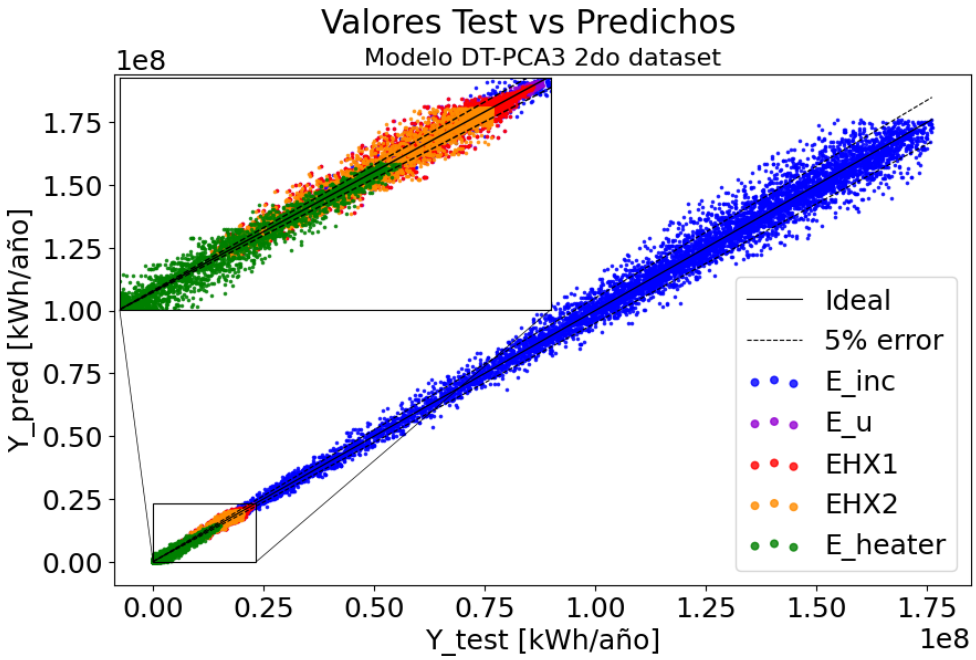


Figura H.3: Valores test vs predichos modelo DT-PCA3 con segundo dataset

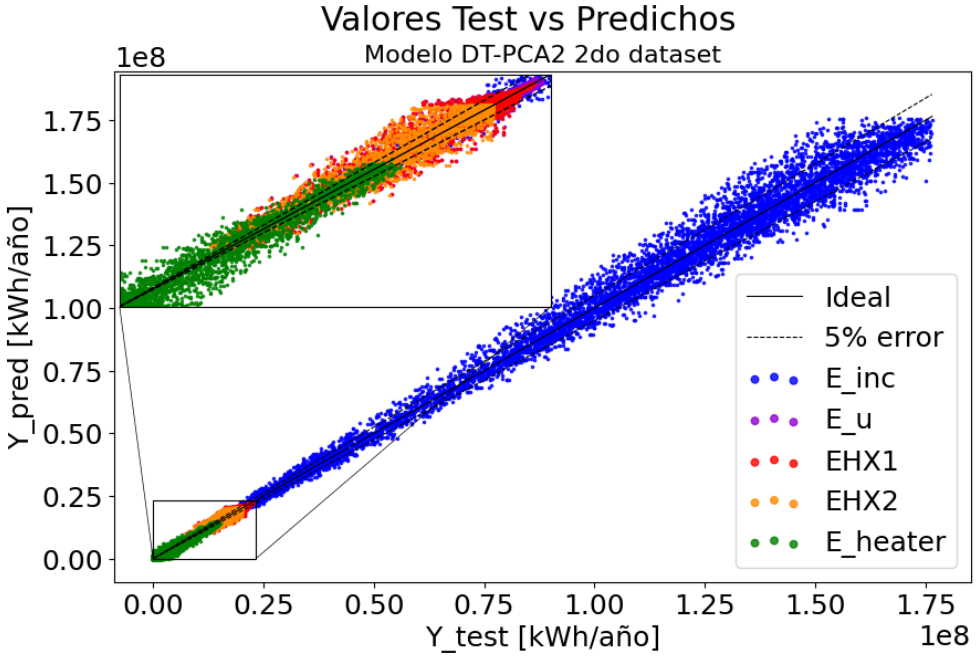


Figura H.4: Valores test vs predichos modelo DT-PCA2 con segundo dataset

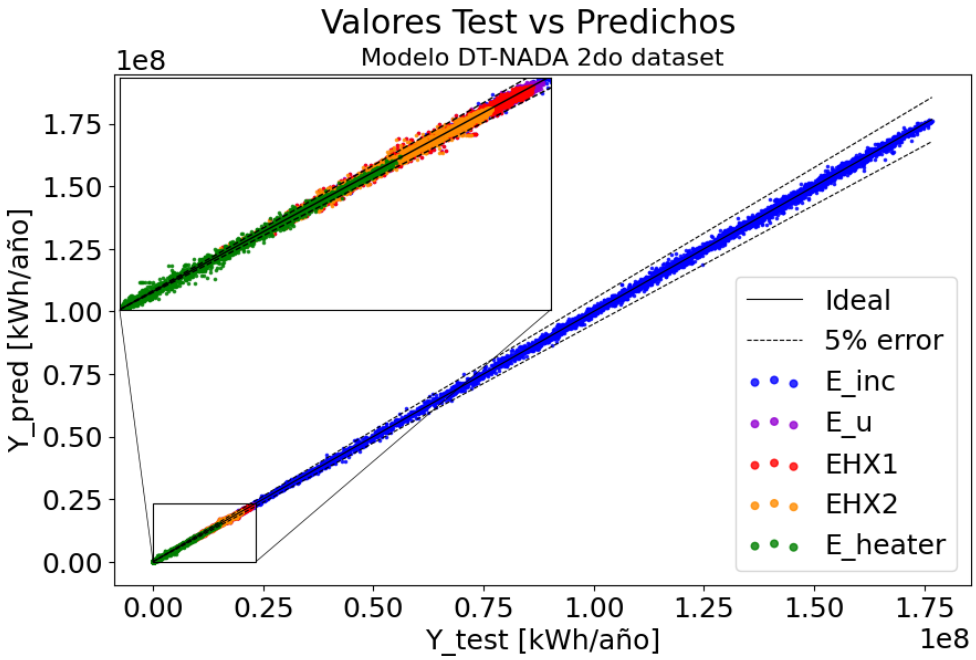


Figura H.5: Valores test vs predichos modelo DT-NADA con segundo dataset

Anexo I. Frente y óptimos de Pareto para optimización multiobjetivo ANN-SS 1

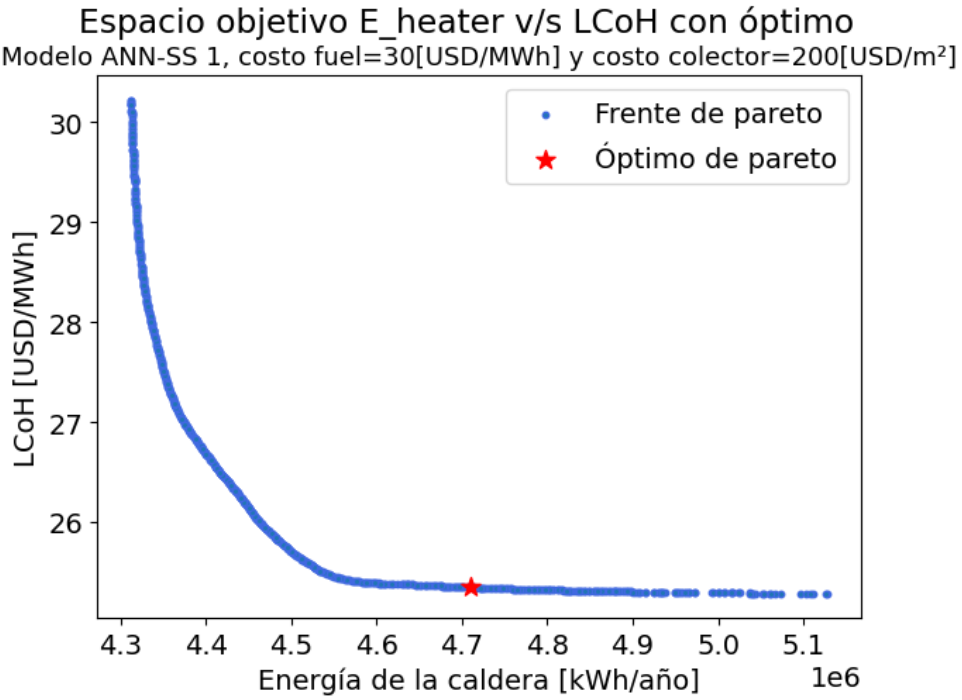


Figura I.1: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=30[USD/MWh] y costo colector=200[USD/m^2]

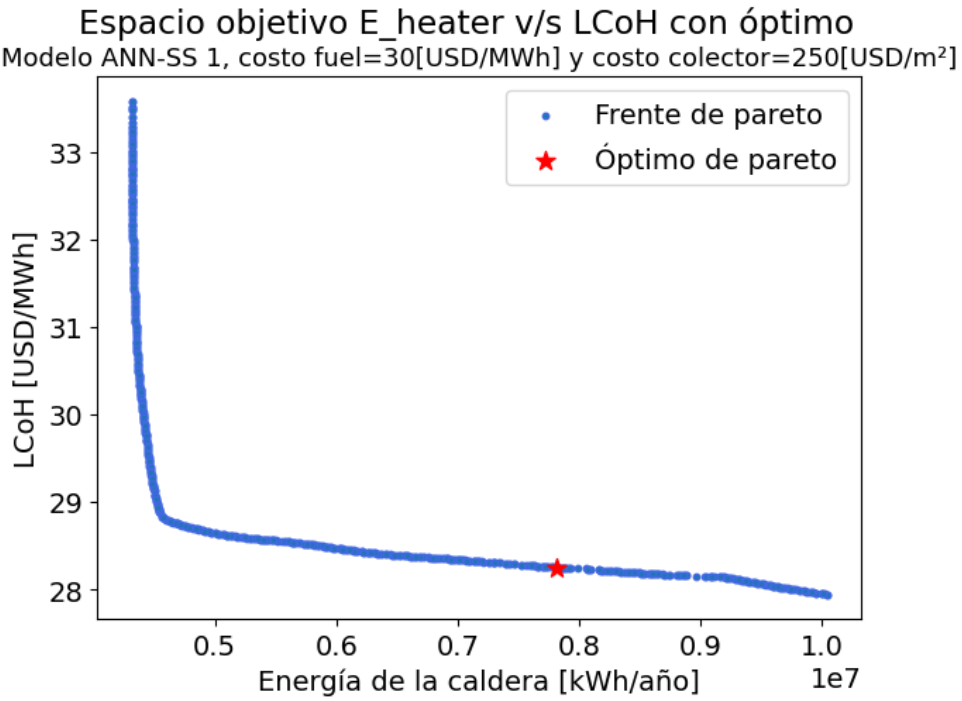


Figura I.2: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=30[USD/MWh] y costo colector=250[USD/m^2]

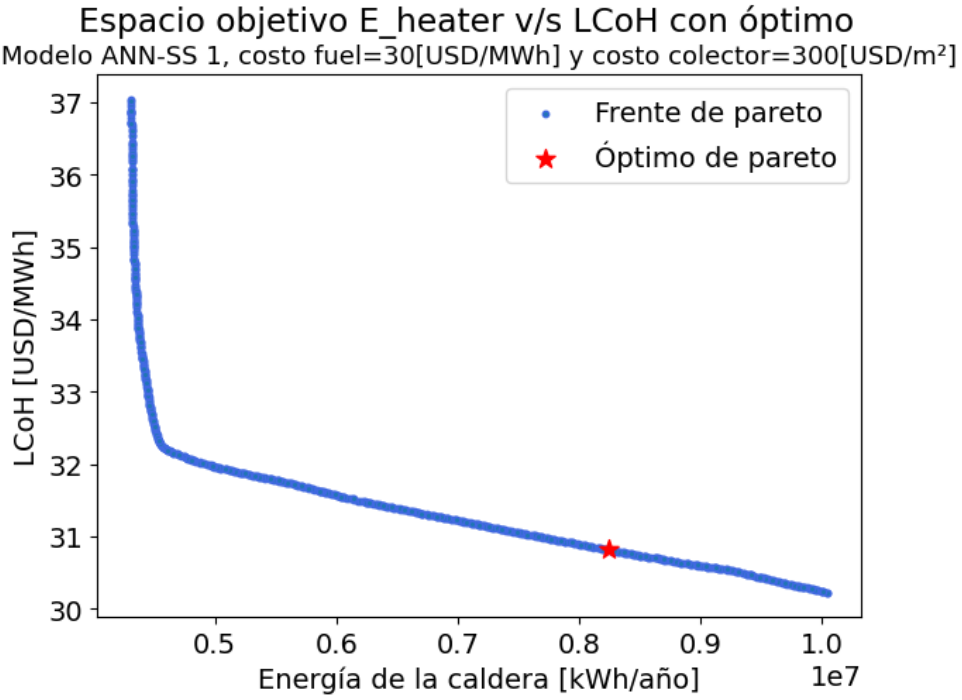


Figura I.3: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=30[USD/MWh] y costo colector=300[USD/m^2]

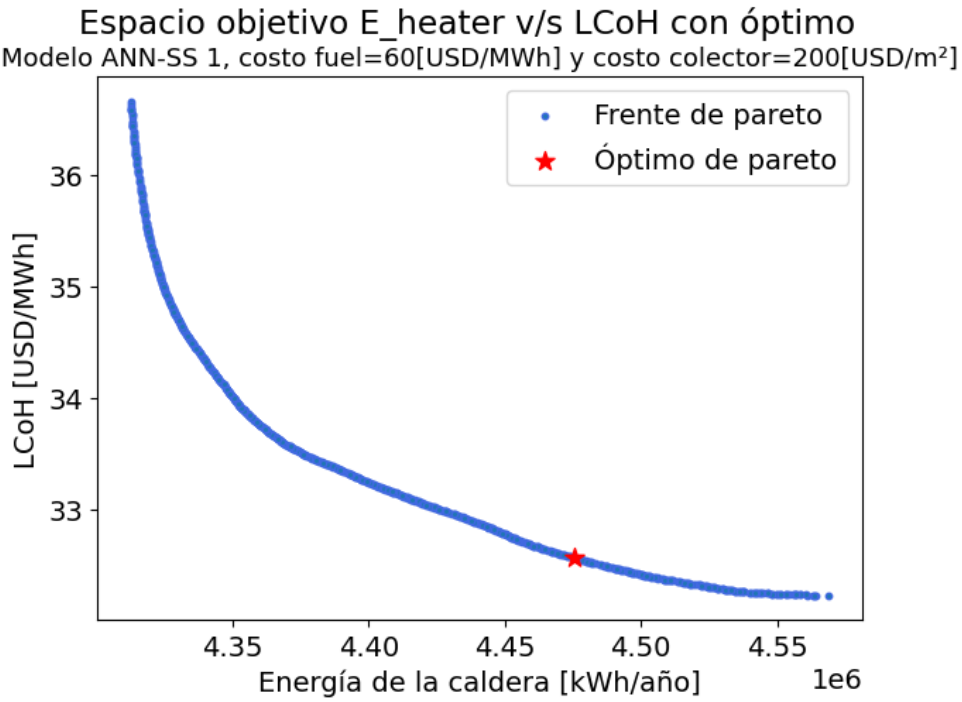


Figura I.4: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=60[USD/MWh] y costo colector=200[USD/m^2]

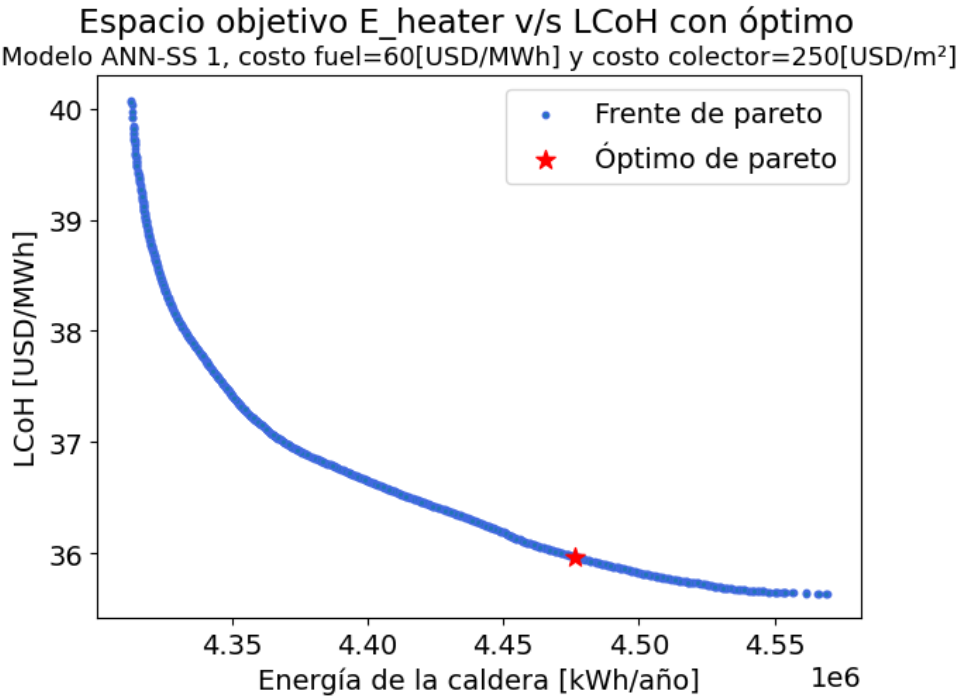


Figura I.5: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=60[USD/MWh] y costo colector=250[USD/m^2]

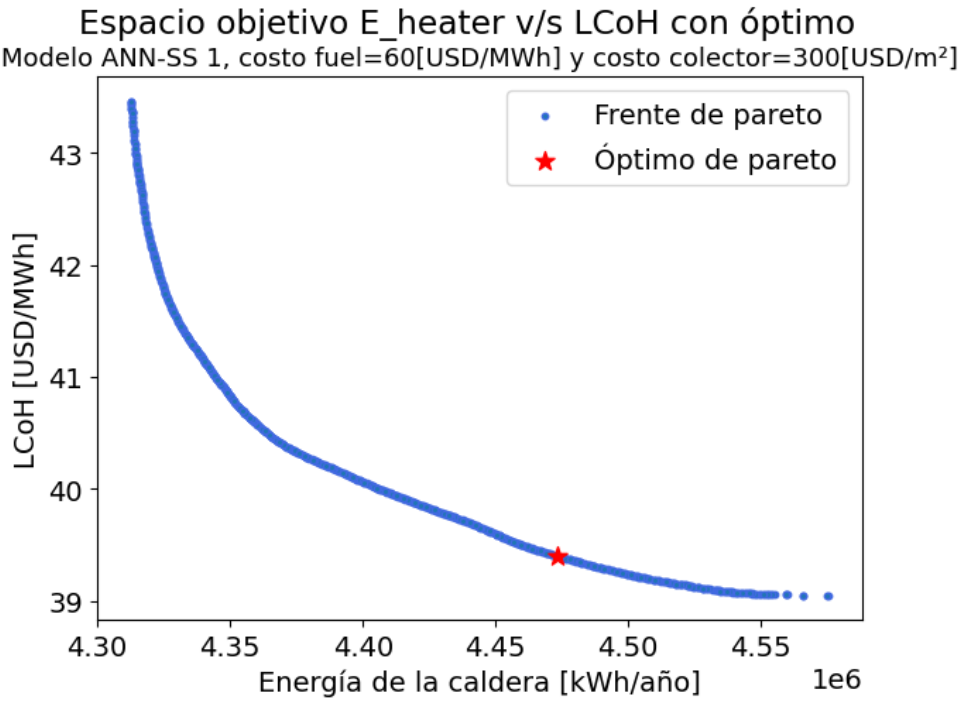


Figura I.6: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=60[USD/MWh] y costo colector=300[USD/m^2]

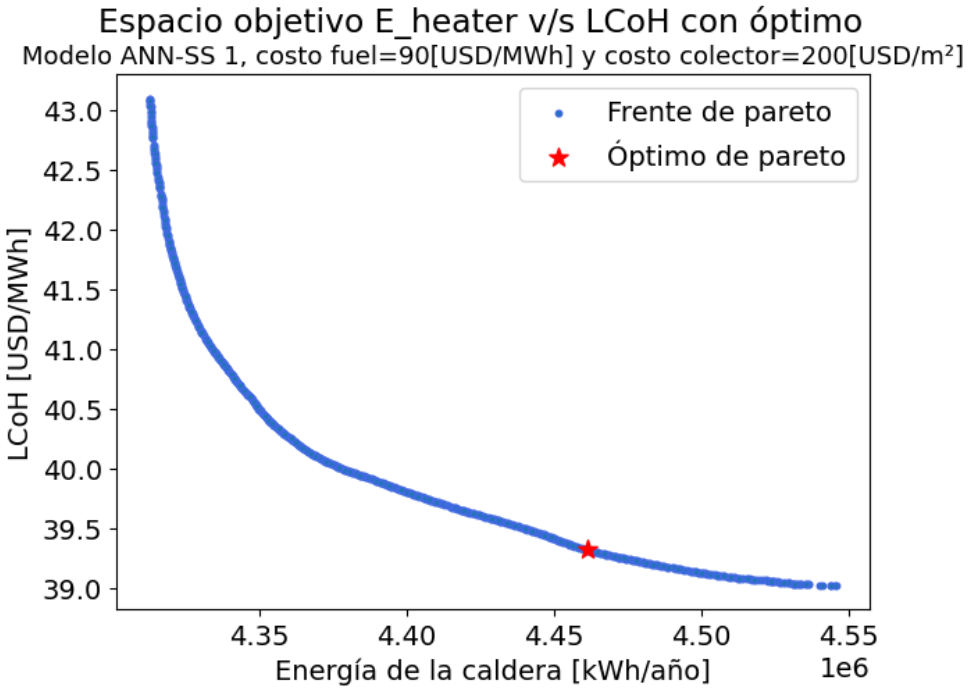


Figura I.7: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=90[USD/MWh] y costo colector=200[USD/m^2]

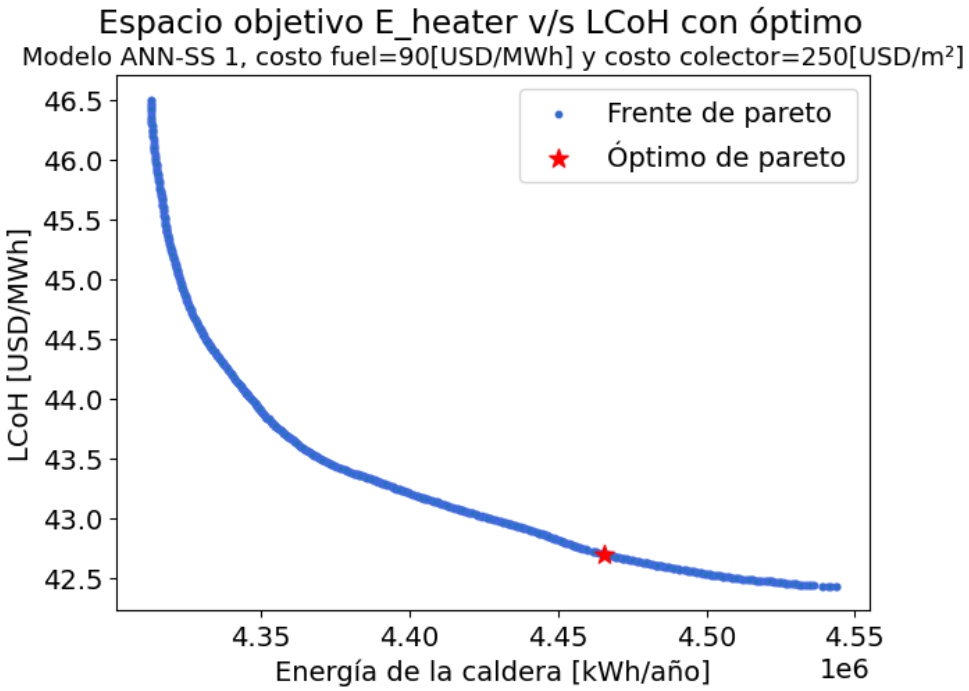


Figura I.8: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=90[USD/MWh] y costo colector=250[USD/m^2]

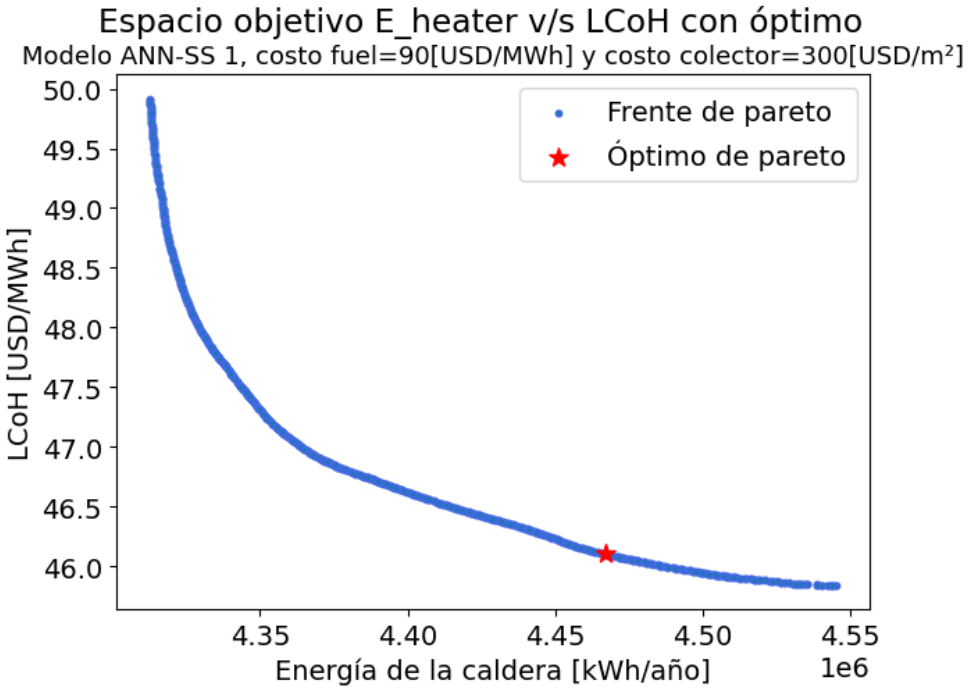


Figura I.9: Frente y óptimo de Pareto ANN-SS 1 con costo fuel=90[USD/MWh] y costo colector=300[USD/m^2]

Anexo J. Frente y óptimos de Pareto para optimización multiobjetivo DT-SS 1

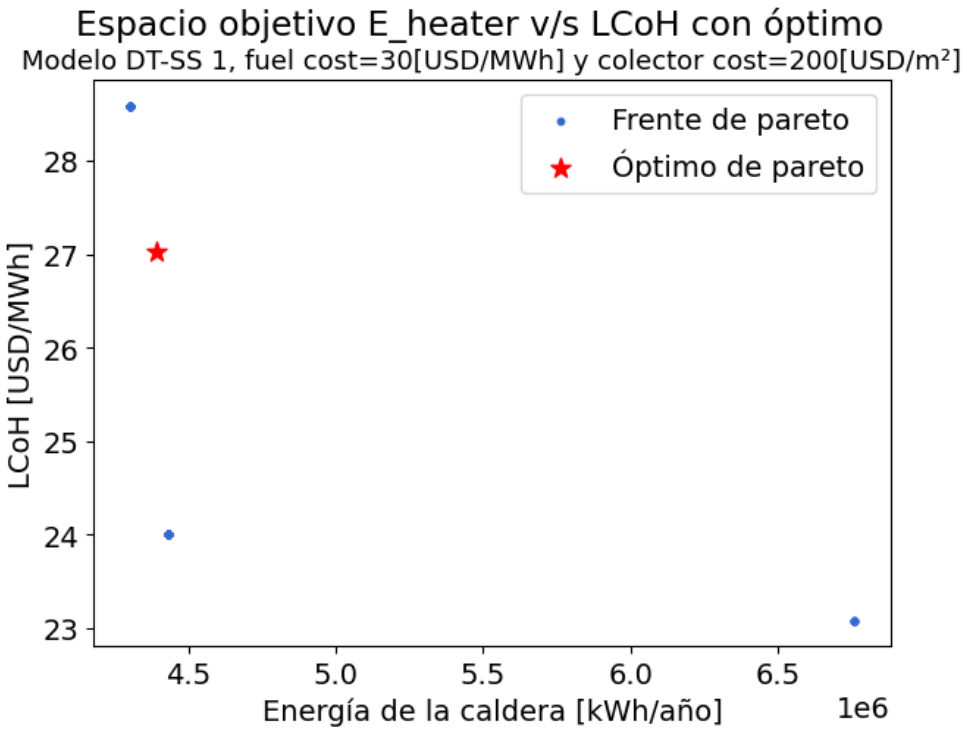


Figura J.1: Frente y óptimo de Pareto DT-SS 1 con costo fuel=30[USD/MWh] y costo colector=200[USD/m^2]

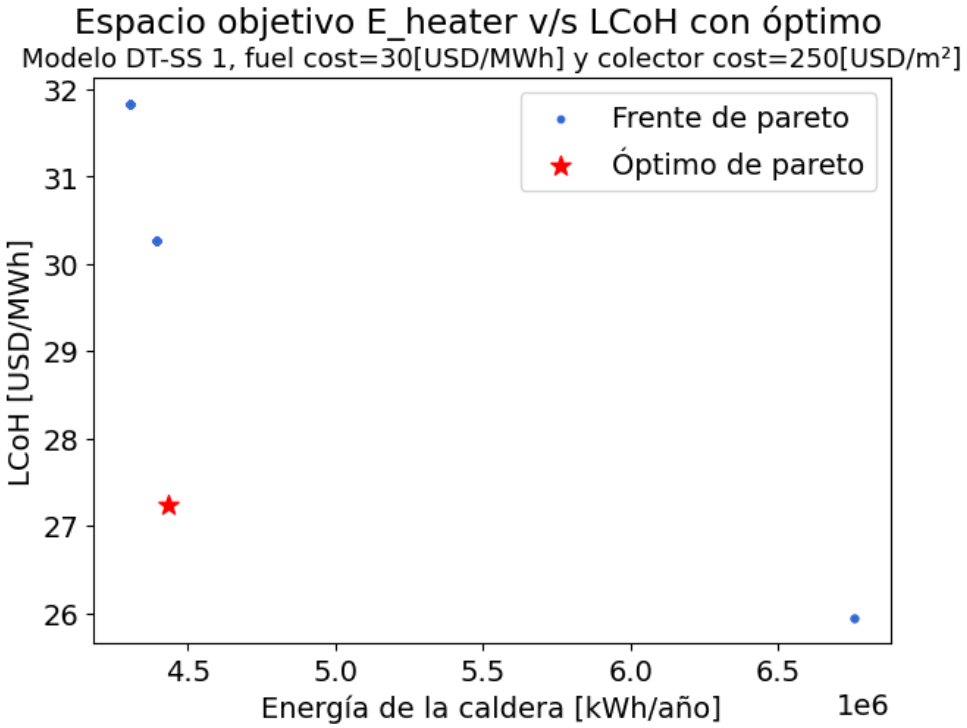


Figura J.2: Frente y óptimo de Pareto DT-SS 1 con costo fuel=30[USD/MWh] y costo colector=250[USD/m^2]

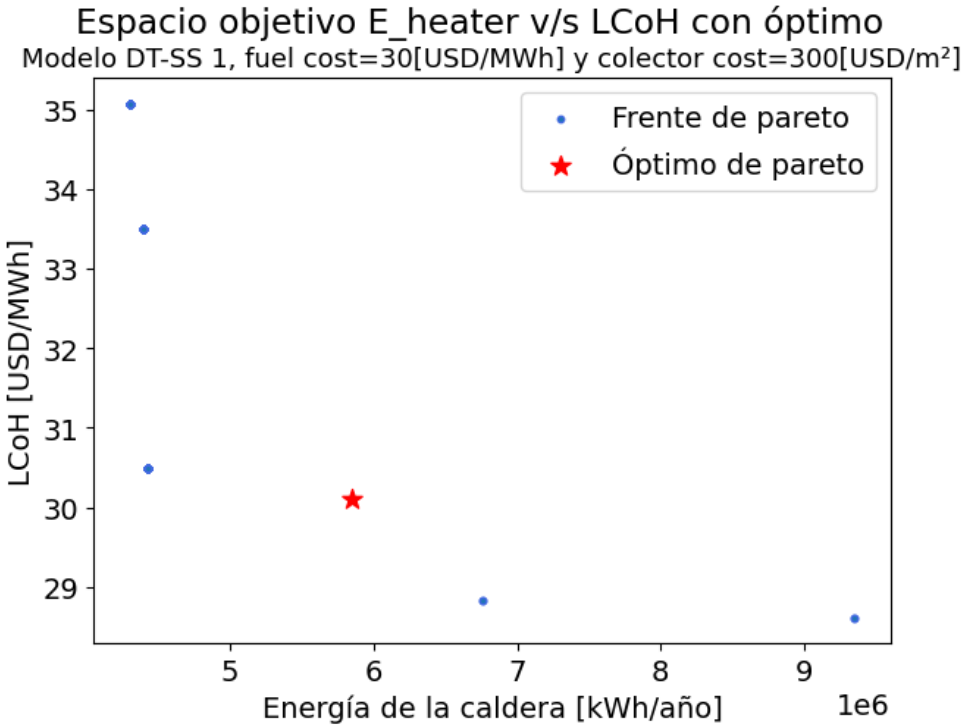


Figura J.3: Frente y óptimo de Pareto DT-SS 1 con costo fuel=30[USD/MWh] y costo colector=300[USD/m^2]

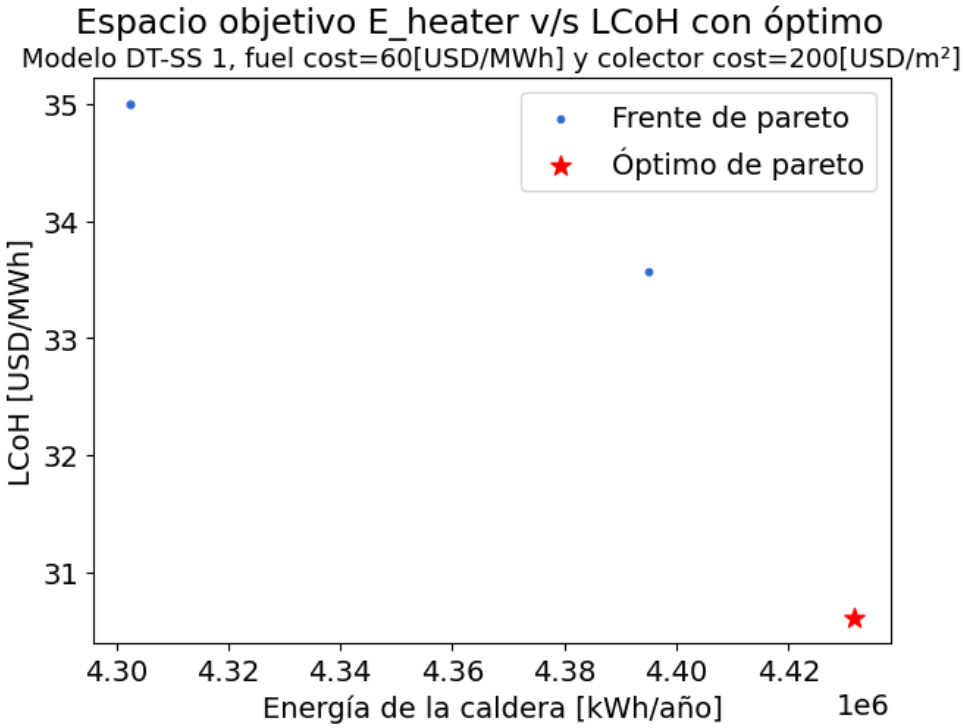


Figura J.4: Frente y óptimo de Pareto DT-SS 1 con costo fuel=60[USD/MWh] y costo colector=200[USD/m^2]

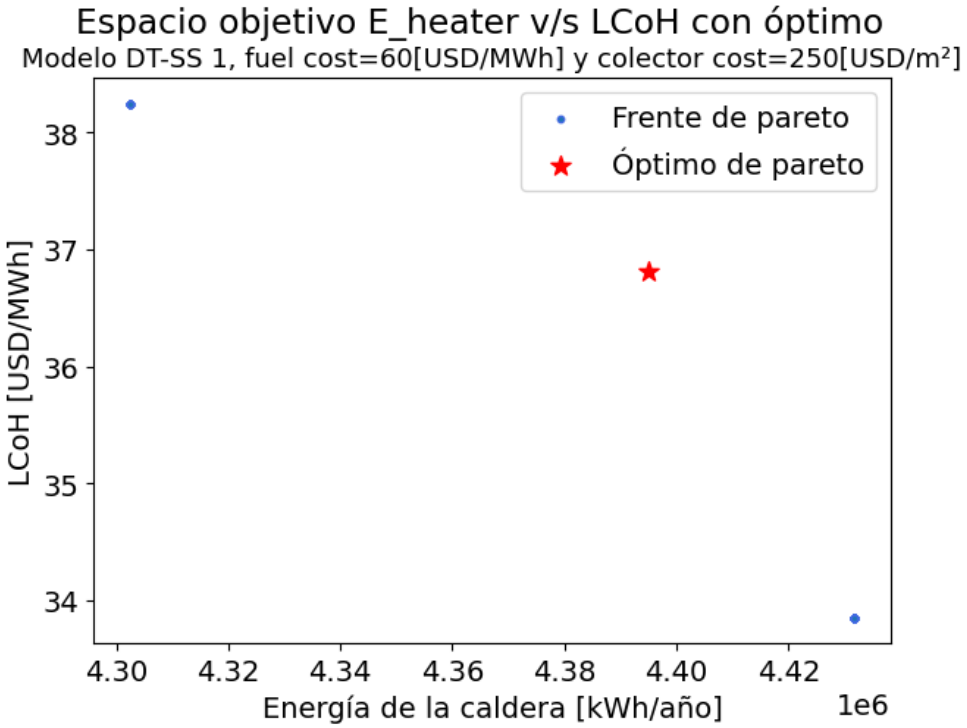


Figura J.5: Frente y óptimo de Pareto DT-SS 1 con costo fuel=60[USD/MWh] y costo colector=250[USD/m^2]

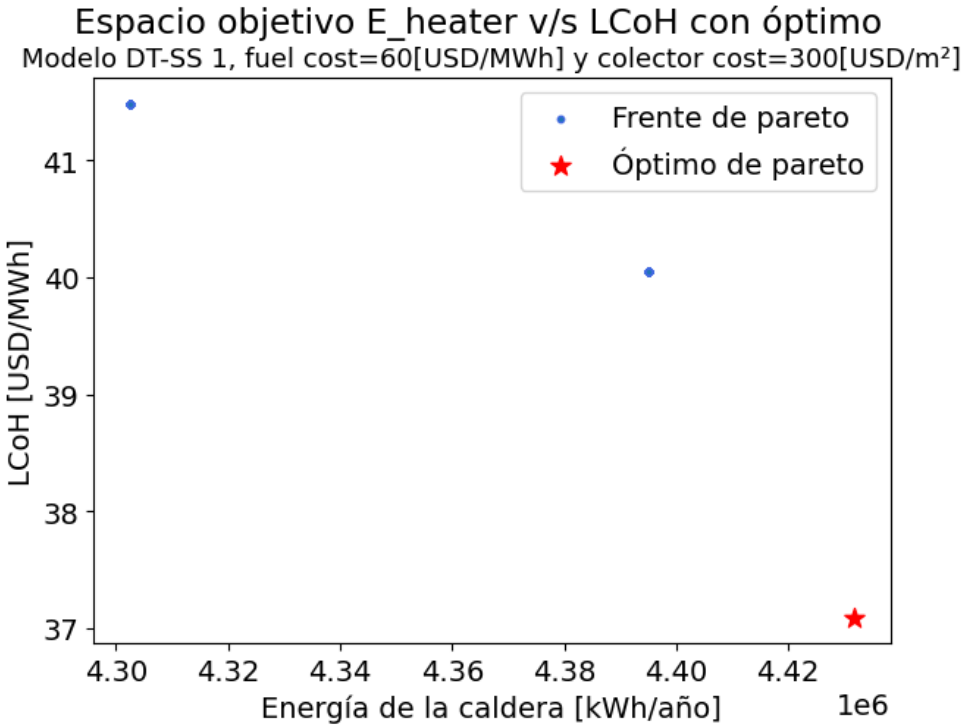


Figura J.6: Frente y óptimo de Pareto DT-SS 1 con costo fuel=60[USD/MWh] y costo colector=300[USD/m^2]

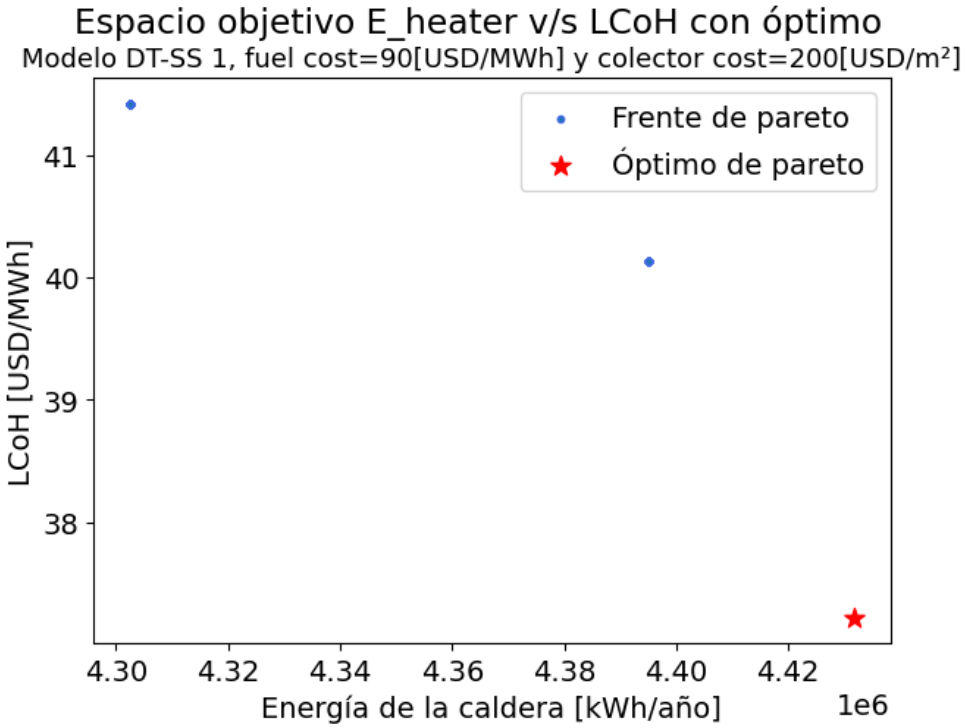


Figura J.7: Frente y óptimo de Pareto DT-SS 1 con costo fuel=90[USD/MWh] y costo colector=200[USD/m^2]

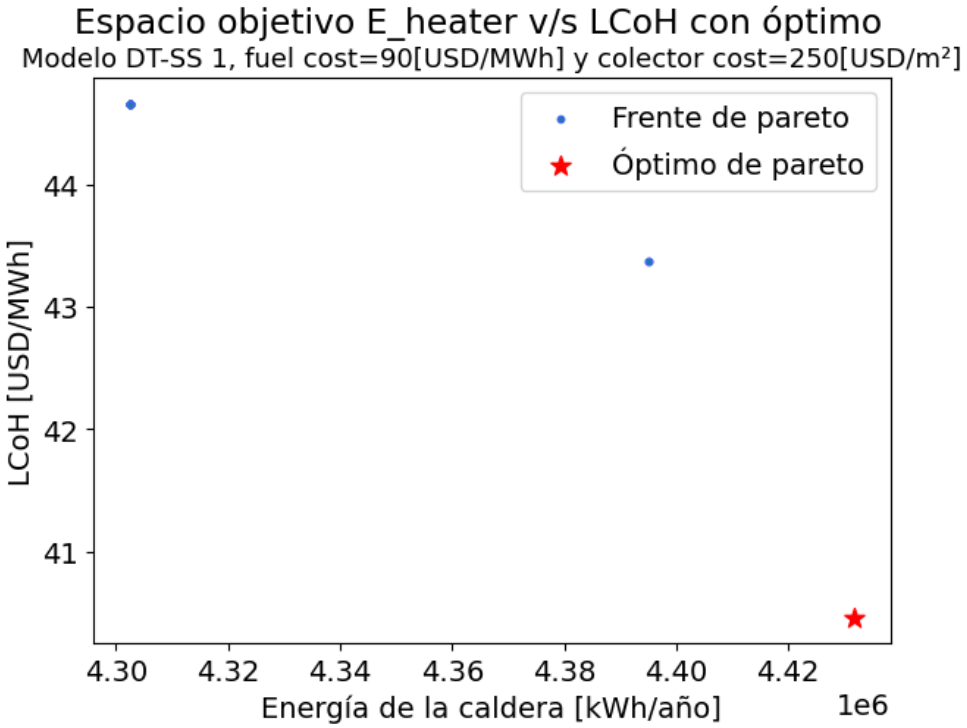


Figura J.8: Frente y óptimo de Pareto DT-SS 1 con costo fuel=90[USD/MWh] y costo colector=250[USD/m^2]

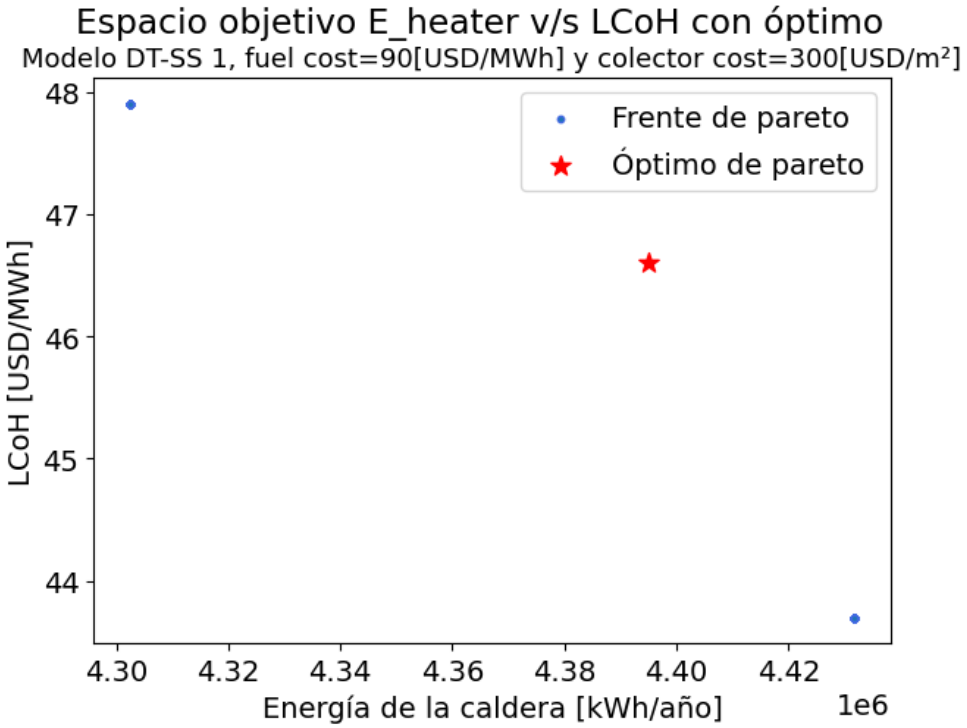


Figura J.9: Frente y óptimo de Pareto DT-SS 1 con costo fuel=90[USD/MWh] y costo colector=300[USD/m^2]

Anexo K. Frente y óptimos de Pareto para optimización multiobjetivo ANN-SS 2

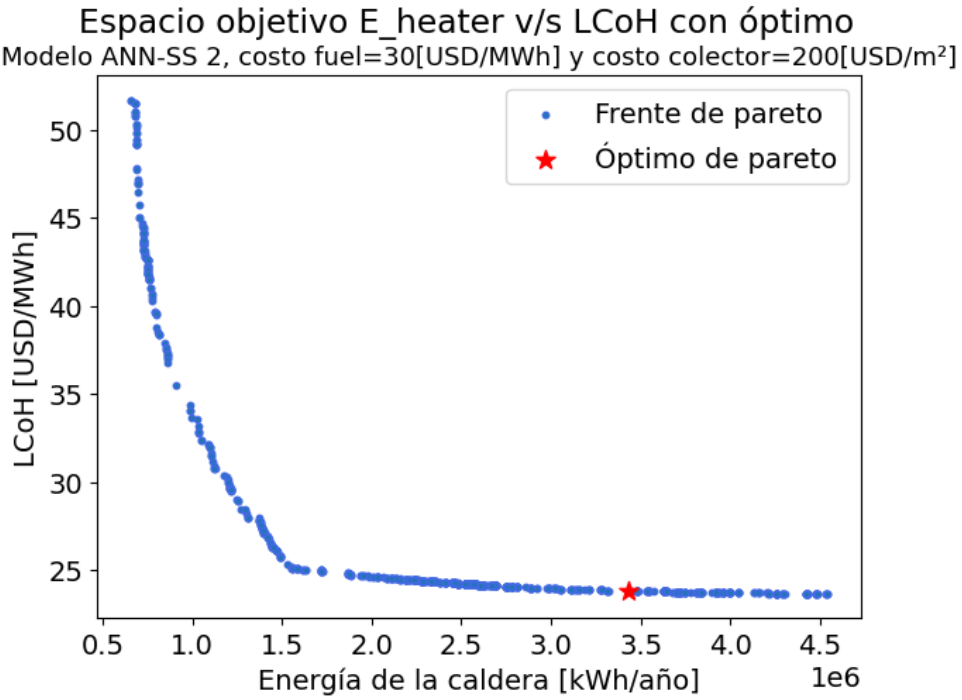


Figura K.1: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=30[USD/MWh] y costo colector=200[USD/m^2]

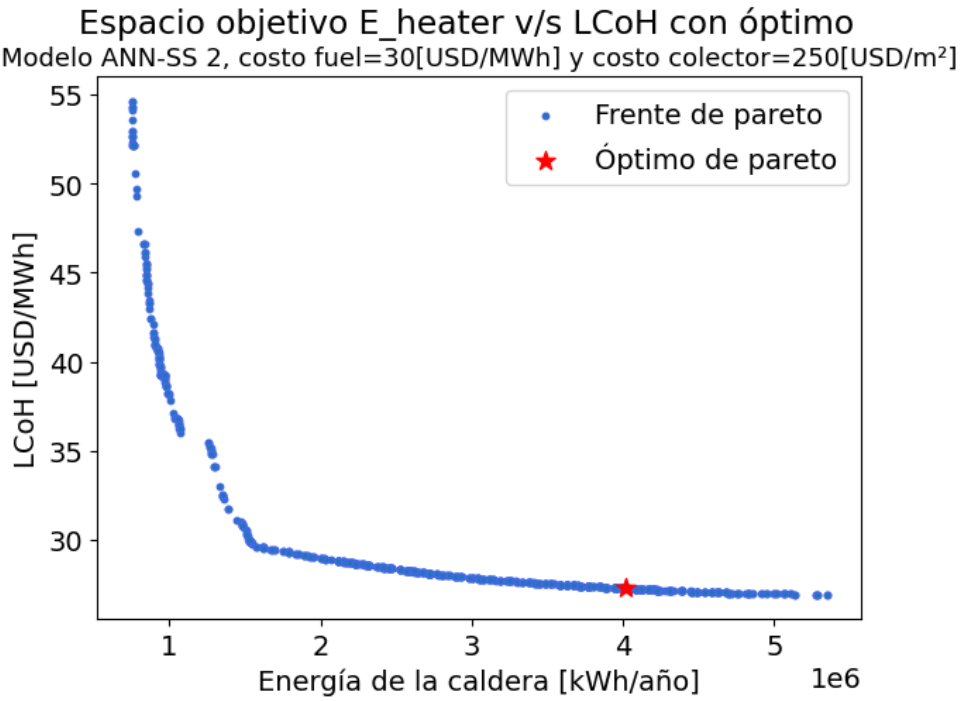


Figura K.2: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=30[USD/MWh] y costo colector=250[USD/m^2]

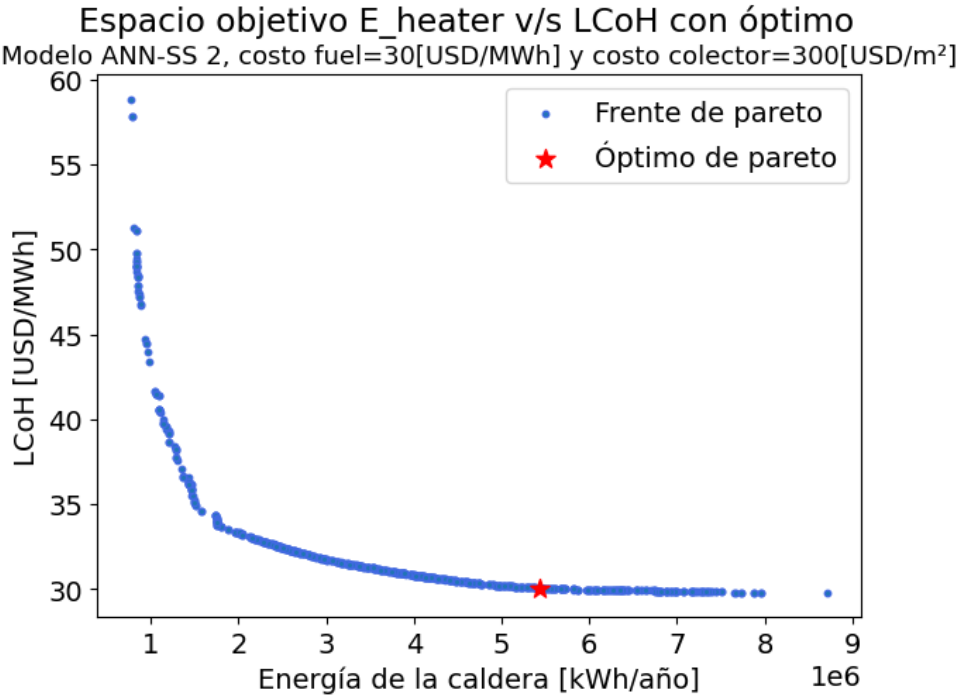


Figura K.3: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=30[USD/MWh] y costo colector=300[USD/m^2]

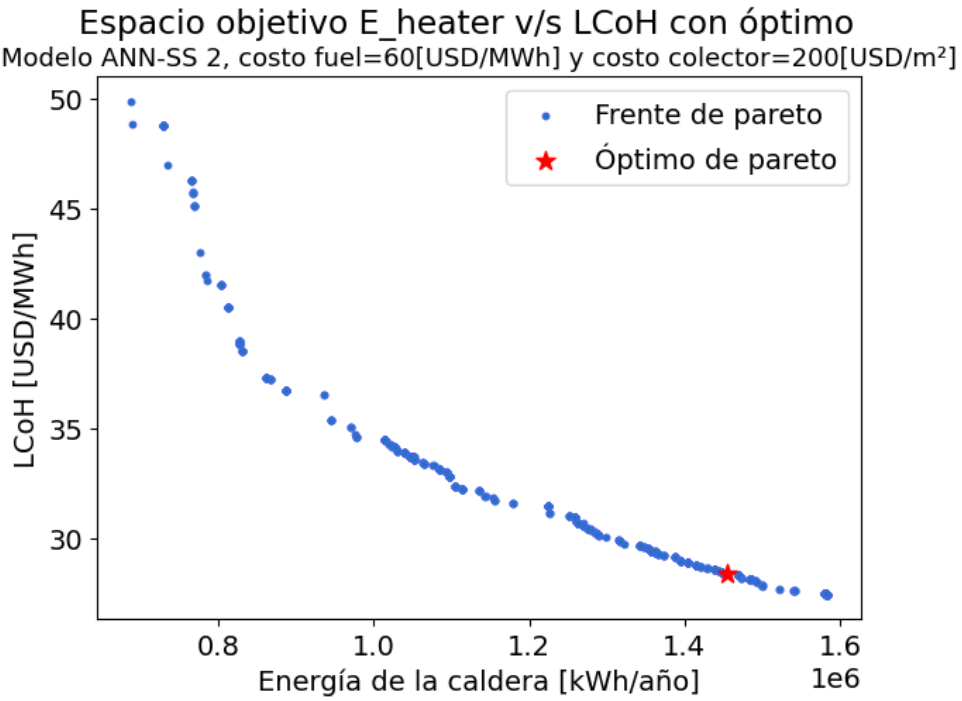


Figura K.4: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=60[USD/MWh] y costo colector=200[USD/m^2]

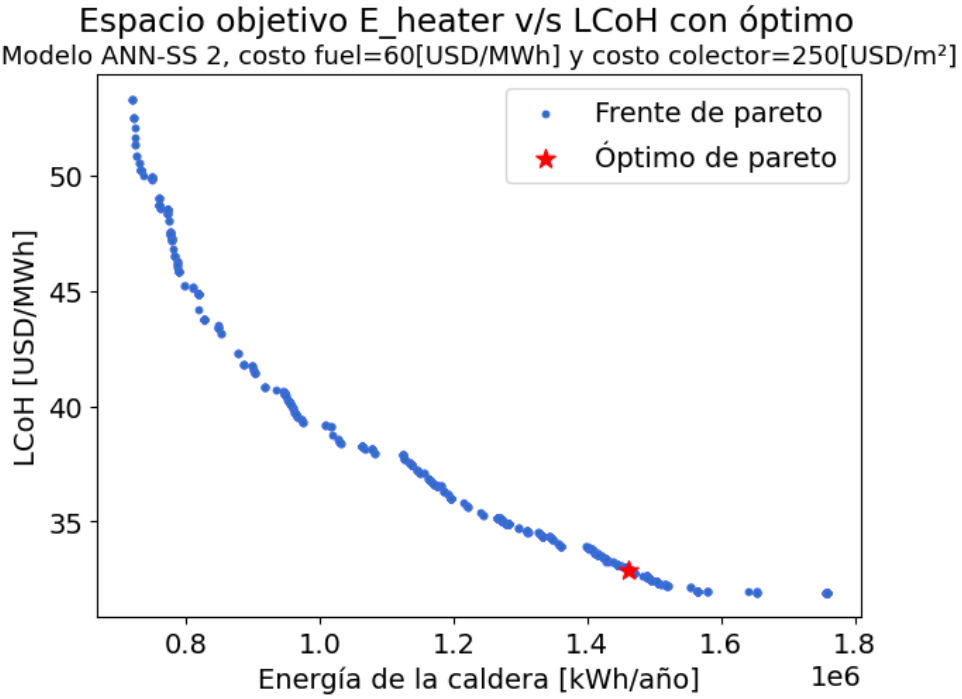


Figura K.5: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=60[USD/MWh] y costo colector=250[USD/m^2]

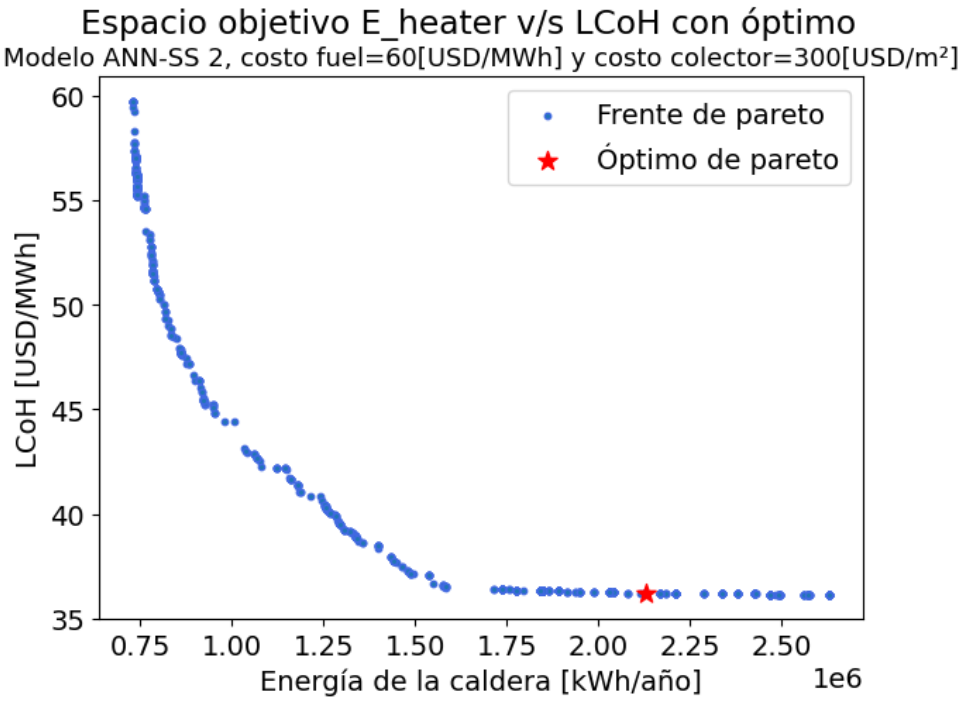


Figura K.6: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=60[USD/MWh] y costo colector=300[USD/m^2]

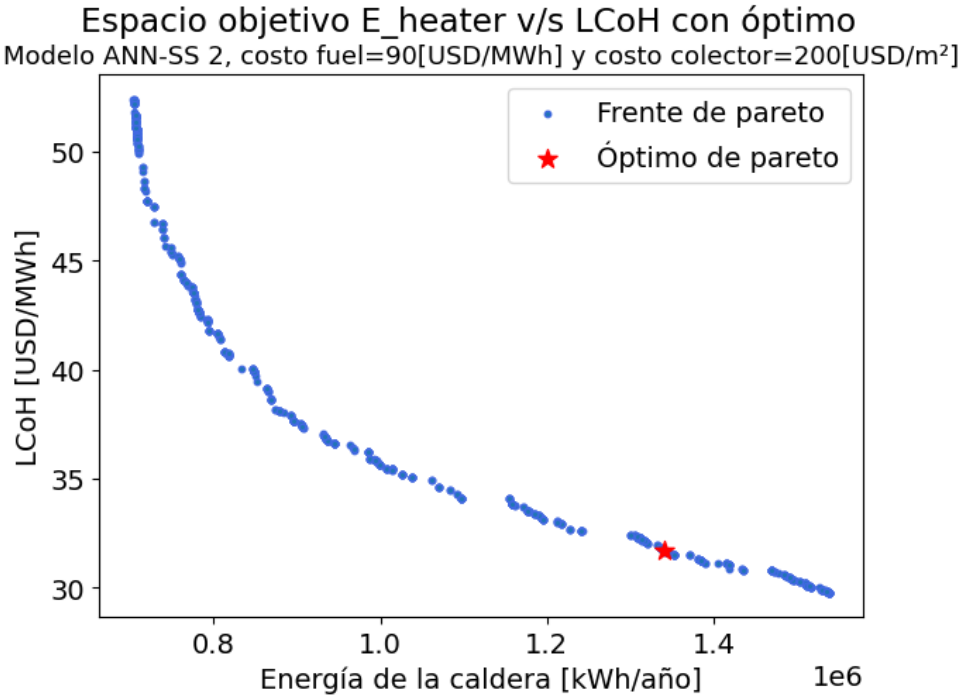


Figura K.7: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=90[USD/MWh] y costo colector=200[USD/m^2]

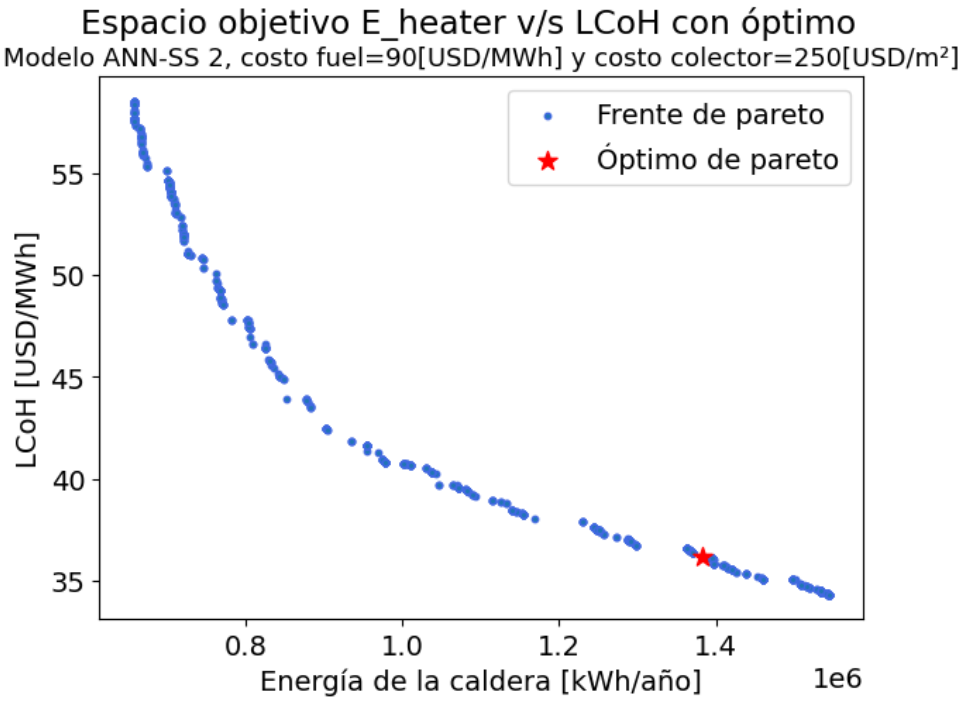


Figura K.8: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=90[USD/MWh] y costo colector=250[USD/m^2]

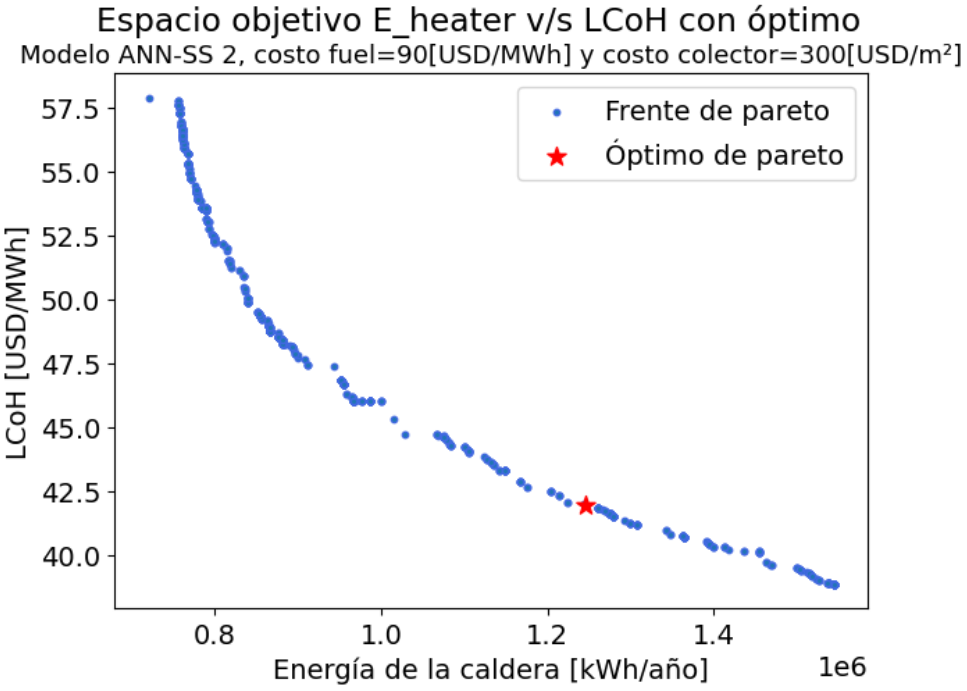


Figura K.9: Frente y óptimo de Pareto ANN-SS 2 con costo fuel=90[USD/MWh] y costo colector=300[USD/m^2]

Anexo L. Frente y óptimos de Pareto para optimización multiobjetivo DT-SS 2

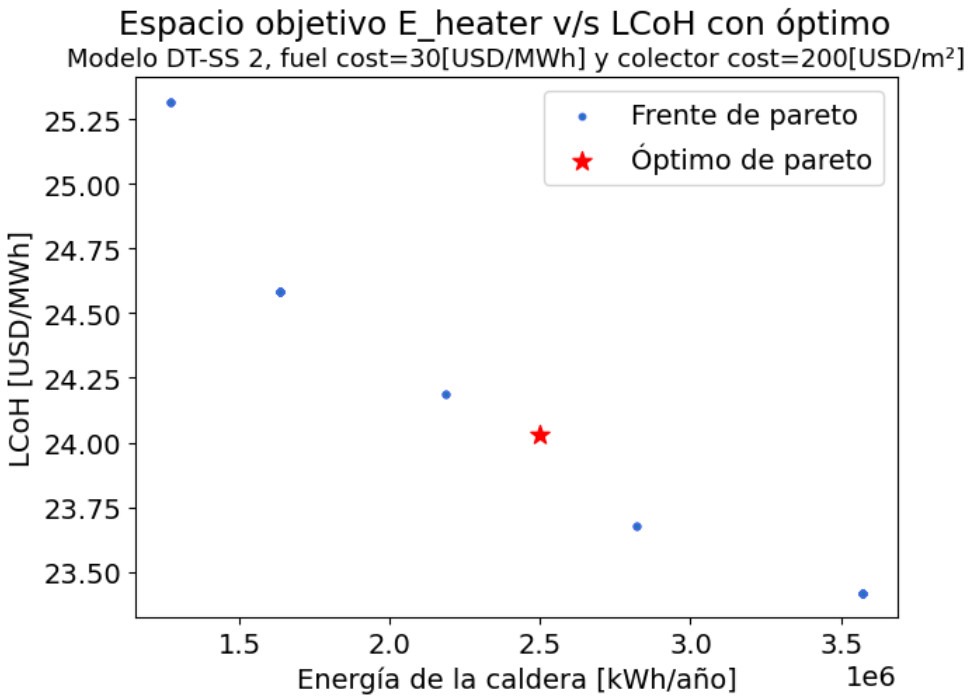


Figura L.1: Frente y óptimo de Pareto DT-SS 2 con costo fuel=30[USD/MWh] y costo colector=200[USD/m^2]

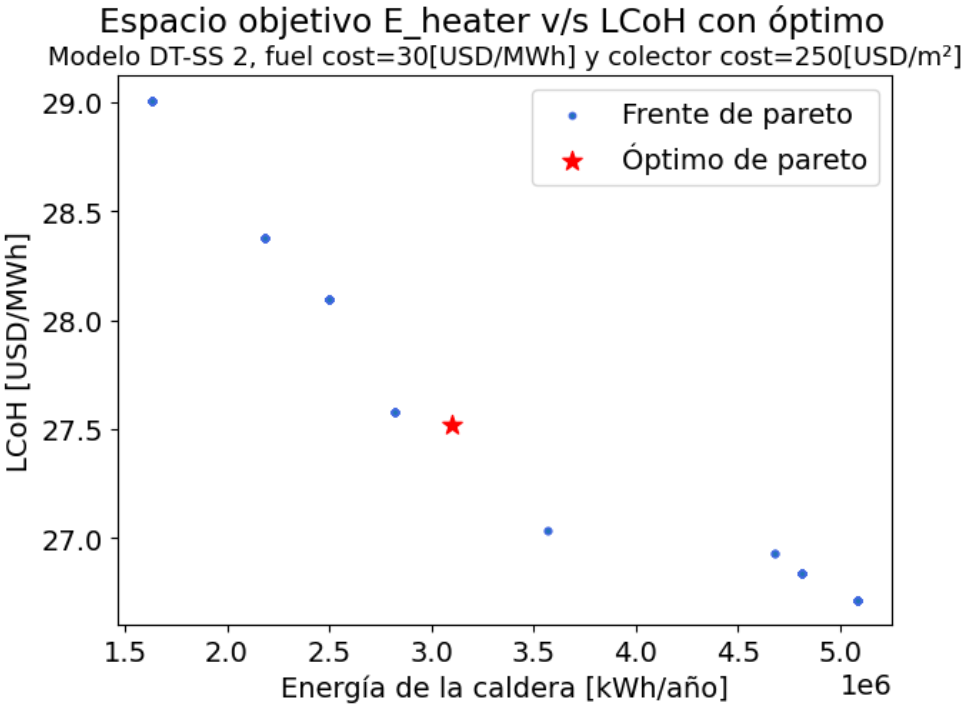


Figura L.2: Frente y óptimo de Pareto DT-SS 2 con costo fuel=30[USD/MWh] y costo colector=250[USD/m^2]

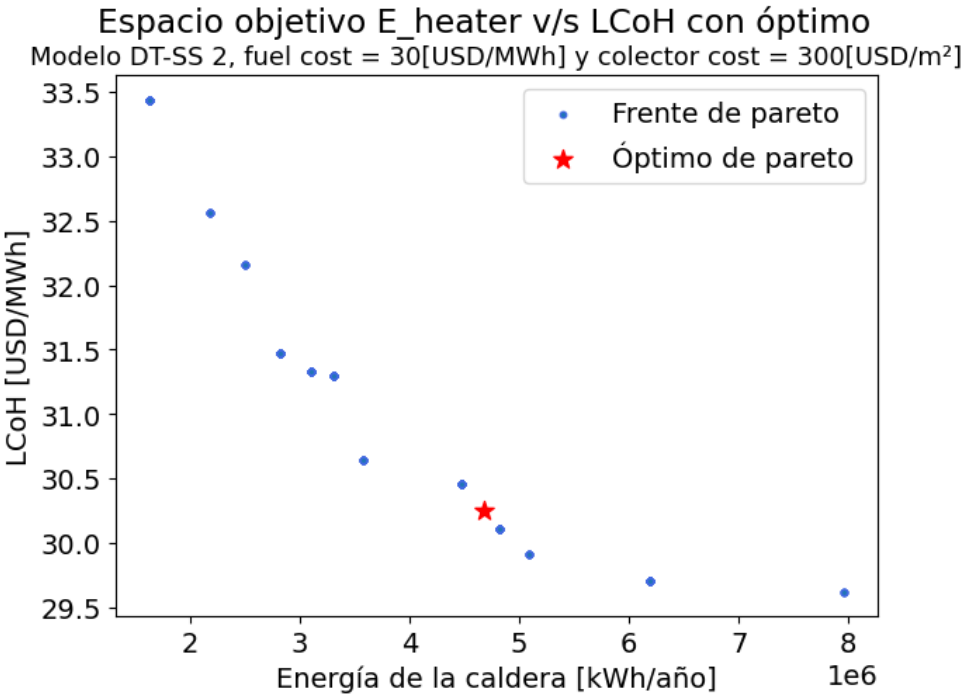


Figura L.3: Frente y óptimo de Pareto DT-SS 2 con costo fuel=30[USD/MWh] y costo colector=300[USD/m^2]

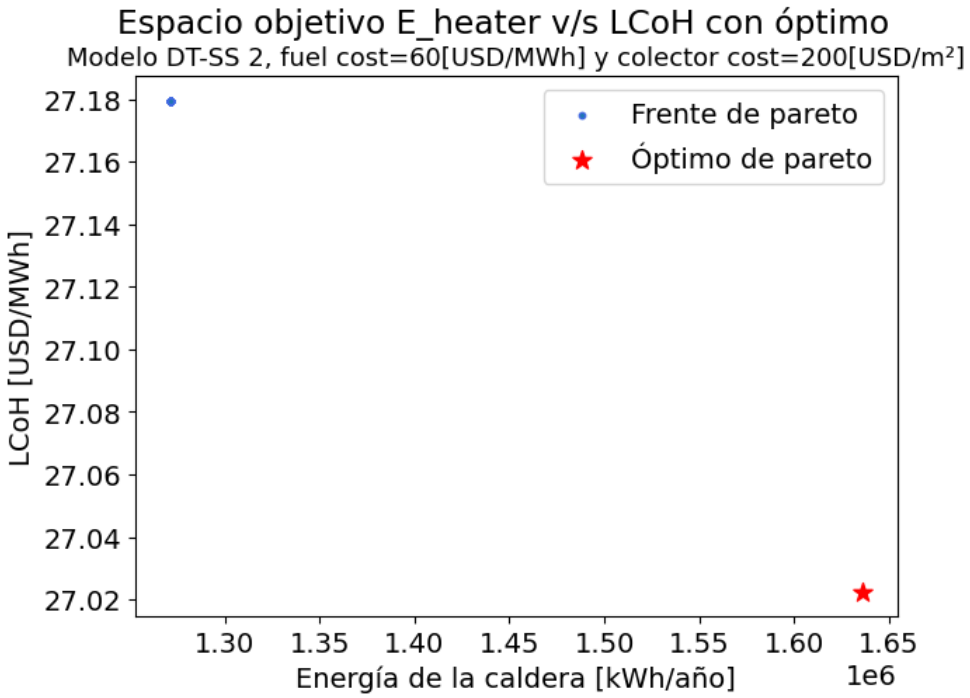


Figura L.4: Frente y óptimo de Pareto DT-SS 2 con costo fuel=60[USD/MWh] y costo colector=200[USD/m^2]

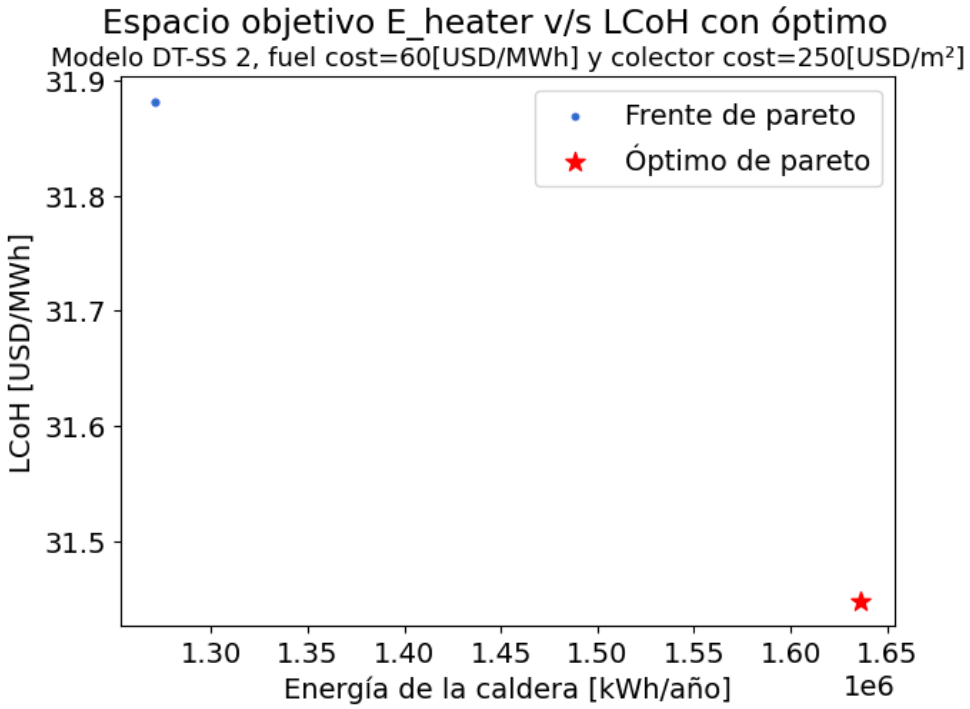


Figura L.5: Frente y óptimo de Pareto DT-SS 2 con costo fuel=60[USD/MWh] y costo colector=250[USD/m^2]

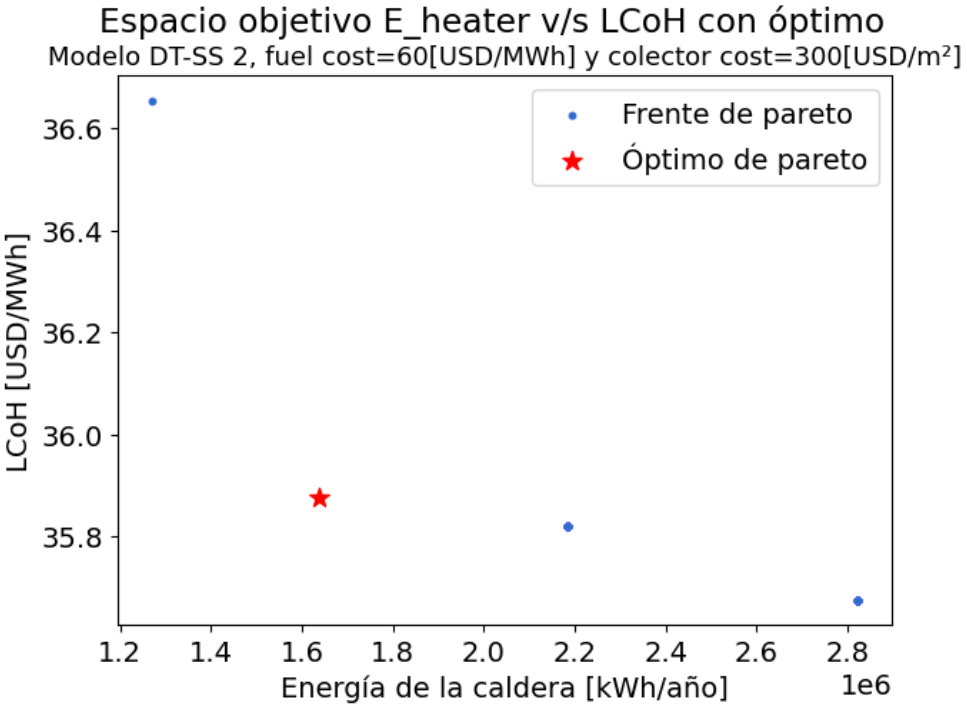


Figura L.6: Frente y óptimo de Pareto DT-SS 2 con costo fuel=60[USD/MWh] y costo colector=300[USD/m^2]

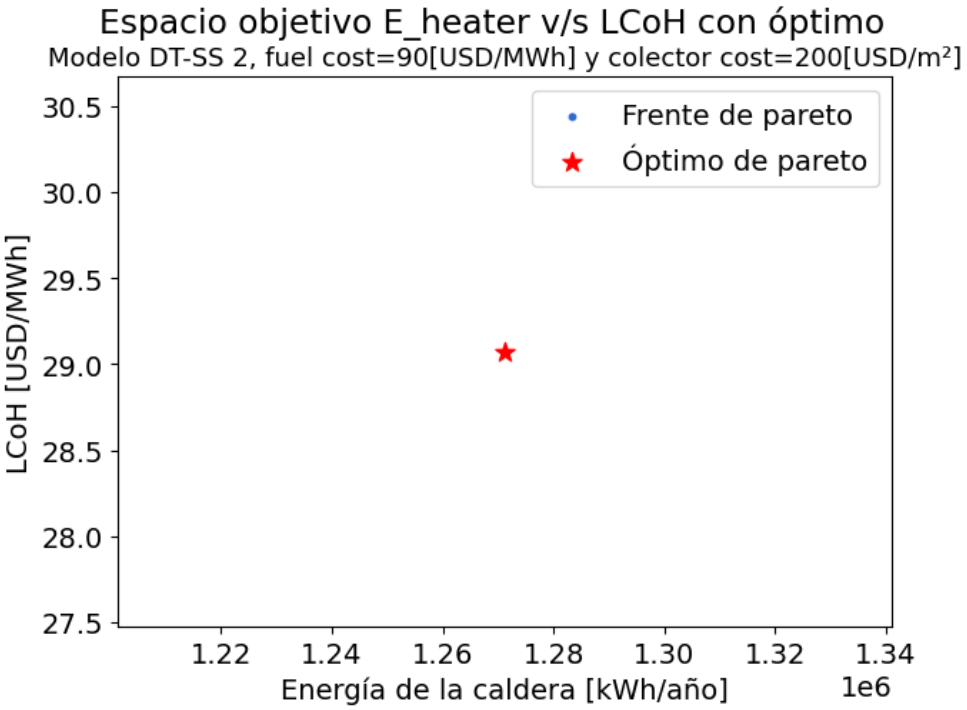


Figura L.7: Frente y óptimo de Pareto DT-SS 2 con costo fuel=90[USD/MWh] y costo colector=200[USD/m^2]

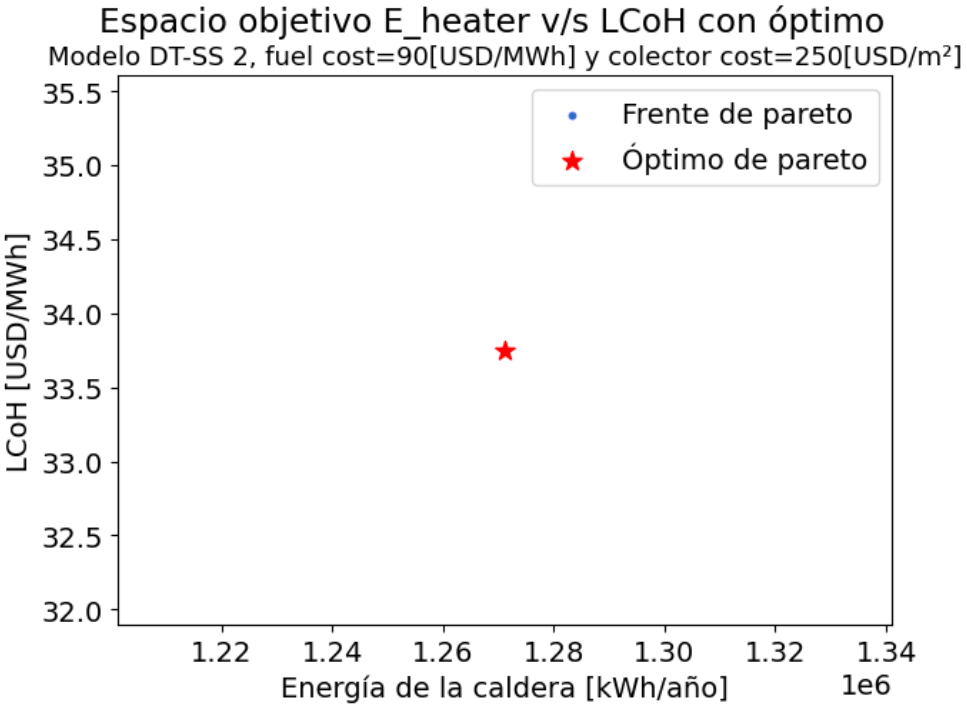


Figura L.8: Frente y óptimo de Pareto DT-SS 2 con costo fuel=90[USD/MWh] y costo colector=250[USD/m^2]

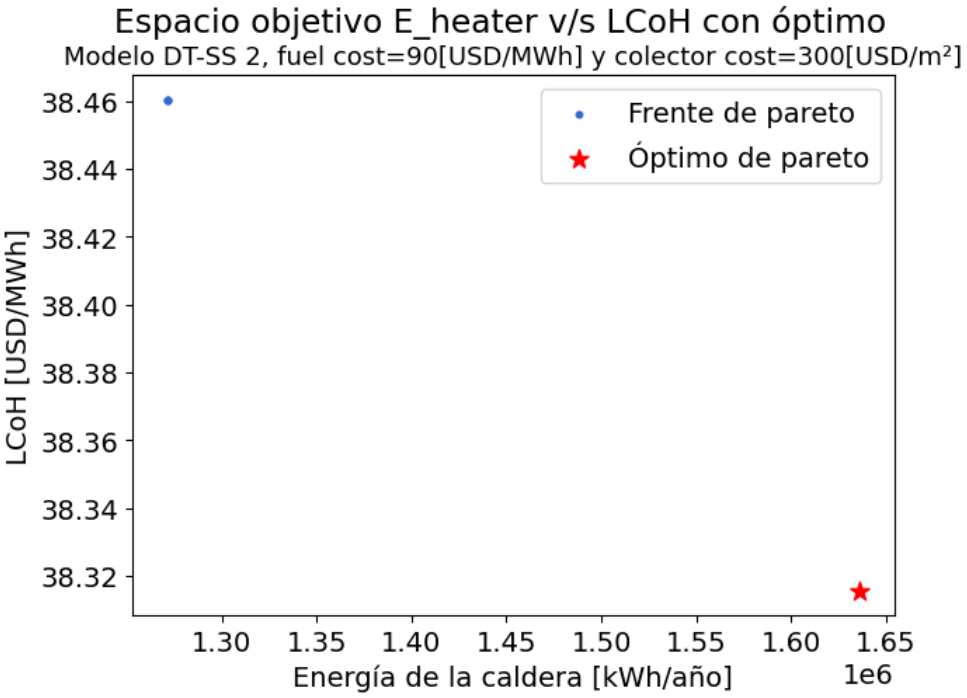


Figura L.9: Frente y óptimo de Pareto DT-SS 2 con costo fuel=90[USD/MWh] y costo colector=300[USD/m^2]