



Informe proyecto pasantía

Predicción de consumo energético en la producción de
alimentos concentrados para bovinos mediante Machine
Learning

Ingeniería civil en energía y medioambiente

Juan José Caldumbide
Alisur
1 de diciembre 2023

Tabla de contenido

| | |
|--|-----------|
| <i>Tabla de ilustraciones</i> | 2 |
| <i>Resumen ejecutivo.....</i> | 3 |
| <i>Executive summary.....</i> | 3 |
| <i>Introducción</i> | 4 |
| <i>Objetivos</i> | 11 |
| <i>Objetivo general</i> | 11 |
| <i>Objetivos específicos</i> | 11 |
| <i>Estado del arte</i> | 11 |
| <i>Soluciones propuestas</i> | 14 |
| <i>Solución a implementar</i> | 15 |
| <i>Riesgos y sus mitigaciones</i> | 17 |
| <i>Evaluación económica.....</i> | 17 |
| <i>Metodologías</i> | 18 |
| <i>Medidas de desempeño</i> | 19 |
| <i>Desarrollo del proyecto basado en la metodología e implementación</i> | 20 |
| <i>Resultados cualitativos y cuantitativos.....</i> | 23 |
| <i>Resultados cuantitativos:</i> | 23 |
| <i>Resultados cualitativos:</i> | 23 |
| <i>Conclusiones y discusión</i> | 23 |
| <i>Referencias.....</i> | 24 |
| <i>Anexo</i> | 25 |

Tabla de ilustraciones

| | |
|---|-----------|
| ILUSTRACIÓN 1: VISTA SATELITAL ALISUR (GOOGLE MAPS)..... | 6 |
| ILUSTRACIÓN 2: ORGANIGRAMA ALISUR | 7 |
| ILUSTRACIÓN 3: PROCESO MOLIENDA..... | 8 |
| ILUSTRACIÓN 4: PROCESO PELETIZADO..... | 8 |
| ILUSTRACIÓN 5: KWH/TON MENSUAL | 10 |
| ILUSTRACIÓN 6: "ESTADO DEL ARTE DE LA PREDICCIÓN DE VARIABLES EN SISTEMAS DE INGENIERÍA ELÉCTRICA BASADA EN INTELIGENCIA ARTIFICIAL" | 13 |
| ILUSTRACIÓN 7: "ESTADO DEL ARTE DE LA PREDICCIÓN DE VARIABLES EN SISTEMAS DE INGENIERÍA ELÉCTRICA BASADA EN INTELIGENCIA ARTIFICIAL" | 14 |
| ILUSTRACIÓN 8: DIAGRAMA DE FLUJO PROGRAMA PROCESAMIENTO DE DATOS | 22 |

Resumen ejecutivo

El presente informe aborda un proyecto innovador en Alisur, una destacada empresa Chilena en el sector de alimentos para la industria salmonera y ganadera. Desde su fundación en 2008, Alisur se ha expandido hacia la alimentación bovina, resaltando por su capacidad de producir alimentos personalizados y disponiendo de una capacidad de producción de 120.000 toneladas anuales.

El proyecto identificó una carencia crucial en el registro de consumos energéticos específicos en los procesos productivos, lo que limitaba la comprensión de los costos exactos de producción para los distintos alimentos producidos. Centrándose en los procesos de molienda y peletizado, los de mayor consumo energético, se desarrolló un modelo predictivo de consumo energético para cada uno de estos procesos, utilizando lecturas de amperaje del software SCADA y datos del área de producción y del área de logística.

El objetivo de este proyecto fue crear un modelo capaz de realizar predicciones del consumo energético basado en la composición y formato del alimento. Este enfoque permitirá a Alisur calcular los costos de producción y evaluar la rentabilidad económica de cada producto. Para lograrlo, se seleccionó el modelo MLPRegressor de Scikit-learn como método principal después de evaluar varias opciones y modelos en el campo del machine learning.

La investigación bibliográfica destaca la originalidad del proyecto, ya que no se encontraron estudios previos que combinaran machine learning con la predicción de consumos energéticos en esta industria. El proyecto implicó un análisis detallado desde el reconocimiento de procesos y equipos hasta la implementación de modelos de aprendizaje automático.

Se espera que este modelo aporte a Alisur una herramienta esencial para la optimización de costos y la toma de decisiones comerciales, marcando un hito significativo en la eficiencia productiva y económica de la empresa.

Executive summary

The present report addresses an innovative project at Alisur, a prominent Chilean company in the food sector serving the salmon and livestock industries. Since its establishment in 2008, Alisur has expanded into cattle feed, standing out for its ability to produce customized feeds with an annual production capacity of 120,000 tons.

The project identified a crucial gap in the specific energy consumption records within the production processes, limiting the understanding of precise production costs for various food products. Focusing on the energy-intensive milling and pelleting processes, a predictive model for energy consumption was developed for each of these processes. This involved utilizing amperage readings from the SCADA software and data from the production and logistics areas.

The objective of this project was to create a model capable of predicting energy consumption based on the composition and format of the food. This approach will enable Alisur to calculate production costs and assess the economic profitability of each product. To achieve this, the MLPRegressor model from Scikit-learn was selected as the primary method after evaluating various options and models in the field of machine learning.

The literature review highlighted the originality of the project, as no previous studies were found that combined machine learning with the prediction of energy consumption in this industry. The project involved a detailed analysis from process and equipment recognition to the implementation of machine learning models.

It is anticipated that this model will provide Alisur with an essential tool for cost optimization and business decision-making, marking a significant milestone in the company's production and economic efficiency.

Introducción

Alisur es una empresa dedicada a procesar y comercializar alimentos para la industria salmonera y ganadera del sur de Chile, sus instalaciones están ubicadas en la comuna de San Pablo, provincia de Osorno, región de Los Lagos.



Ilustración 1: vista aérea ALISUR (PL prensa)

En el año 2008, la empresa comenzó sus operaciones con una planta de extracción de aceite de ráps, producto utilizado como alimento en la industria salmonera. Luego, en el año 2014, se expandieron hacia el rubro de la alimentación bovina, en el cual han alcanzado gran éxito debido a su capacidad de producir alimentos 100% personalizados según los requerimientos nutricionales del cliente. Han llegado a ser la planta más grande de producción de alimentos para la industria ganadera hasta el día de hoy, con una capacidad de producción de 120.000 toneladas al año.



Ilustración 1: vista satelital ALISUR (Google maps)

La empresa cuenta con dos líneas productivas idénticas (10 ton/h cada una) y 100% automatizadas para la producción de alimentos concentrados para vacunos, de la marca Holandesa Ottevanger Milling Engineers. Estas líneas funcionan mediante procesamiento por lotes o 'batch processing', es decir, se programan lotes de producción de 1000 kg mediante el software Batch Explorer en orden secuencial y sin parar, lo que permite fabricar los pedidos de alimentos con distintas proporciones de materias primas sin perder la continuidad.

Luego de una revisión y búsqueda de problemas u oportunidades, se concluyó que existe una falta de datos de consumos energéticos en el proceso productivo de la empresa. No se conocen los costos de producción exactos de los distintos alimentos concentrados producidos, ya que existen múltiples combinaciones de alimentos posibles y cada combinación es definida por el cliente. Como resultado de esto, pueden existir variaciones en el consumo energético de ciertos procesos debido a las distintas propiedades de cada materia prima, como su densidad, su masa, su volumen, etc.

La empresa solo cuenta con los consumos energéticos totales, sin diferenciar qué se está procesando en la planta. Por lo tanto, existe una oportunidad de definir una relación entre cada materia prima, su porcentaje en la fórmula a producir, la presentación deseada y su consumo energético. Con esto, se puede crear un modelo que permita predecir cuál será el consumo energético exacto y así conocer su costo de producción. Esto se hace con el fin de definir qué productos tienen un mayor margen de contribución y así proveer información relevante a la empresa.

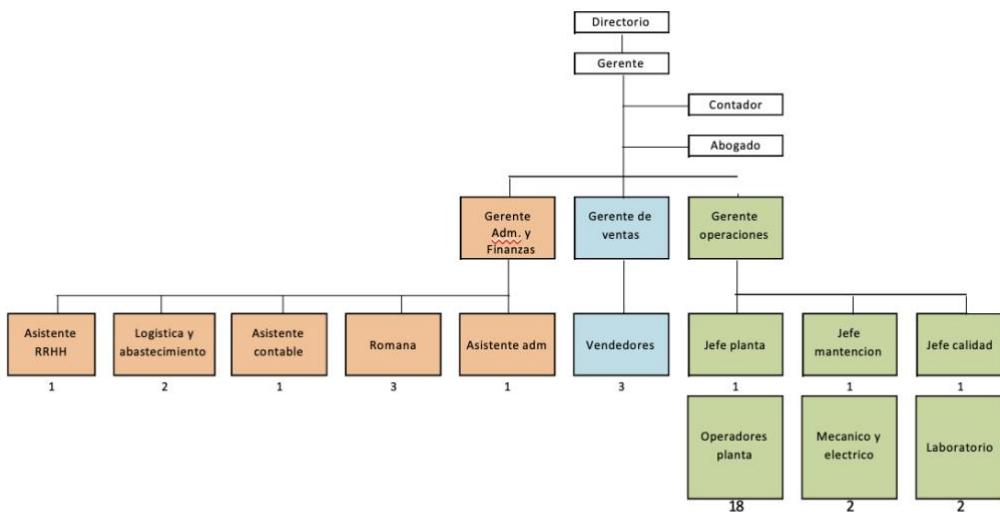


Ilustración 2: organigrama Alisur

El proyecto se desarrollará en el área de producción de la empresa, la cual está bajo supervisión del gerente de operaciones (Ilustración 2organigrama Alisur), y se enfocará en dos procesos específicamente, la molienda (Ilustración 3: proceso molienda Ilustración 4: proceso peletizado) y el peletizado (Ilustración 4), estos son los dos procesos de mayor consumo energético en la planta, los motores de cada proceso tienen una potencia de 90 y 160 kW respectivamente, además son los únicos dos procesos en los cuales existen lecturas de amperaje, las cuales se pueden obtener a través del software SCADA y es información fundamental para llevar a cabo este proyecto.

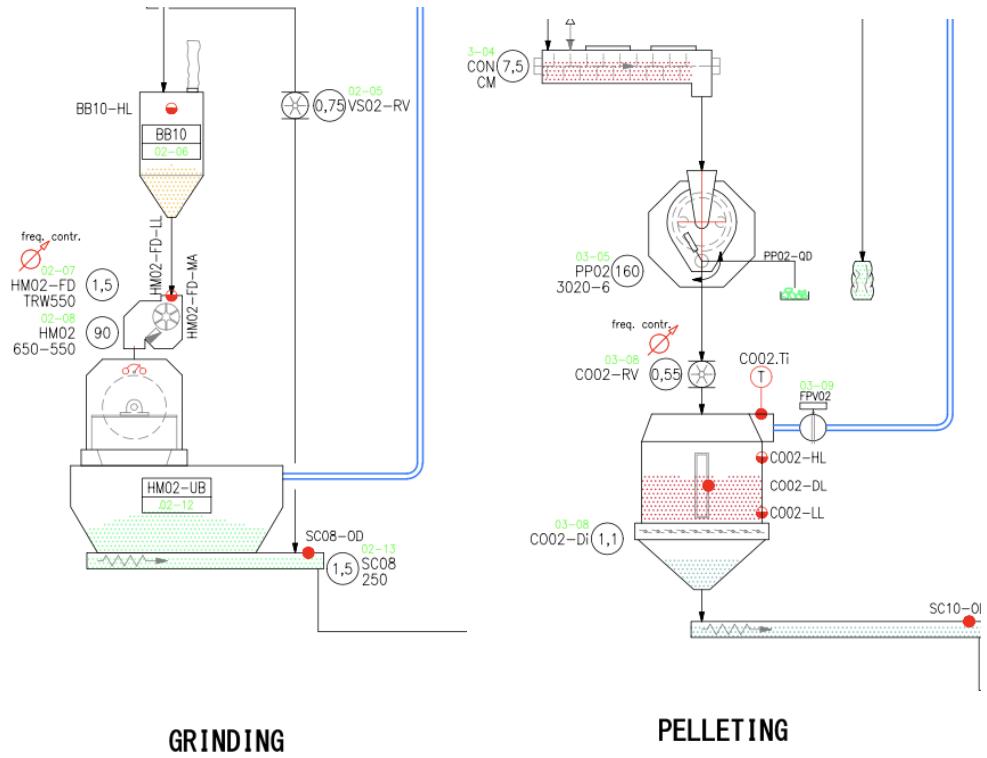


Ilustración 3: proceso molienda

Ilustración 4: proceso peletizado

En la siguiente tabla se detalla la potencia y la cantidad de motores de los procesos principales (molino, mezclador, acondicionador y peletizado), así como del resto de los equipos de la línea productiva tales como transportadores (redler), elevadores, dosificadores, compresor, entre otros. Como se puede evidenciar en la tabla, ninguno de estos motores se acerca a la potencia de los procesos en estudio, 90 y 160 kW. Más del 60% de la potencia de todos los motores de la planta está concentrada en el 4% de estos, es decir, en estos 2 motores. Cabe mencionar que en esta tabla

se incluye el total de motores de la planta, es decir, la suma de los motores de las dos líneas productivas idénticas.

| Proceso | Potencia (kW) | Cantidad |
|----------------|---------------|----------|
| Peletizadora | 160 | 2 |
| Molino | 90 | 2 |
| Mezcladora | 7,5 | 2 |
| Acondicionador | 7,5 | 2 |
| Otros | 0,25 | 2 |
| | 0,55 | 2 |
| | 0,75 | 6 |
| | 1,1 | 3 |
| | 1,2 | 2 |
| | 1,5 | 14 |
| | 2,2 | 46 |
| | 3 | 9 |
| | 4 | 2 |
| | 5,5 | 4 |
| | 7,5 | 2 |
| | 18,5 | 2 |
| Compresor | 25 | 1 |
| Total | 798 | 103 |

Tabla 1: detalle motores planta

La siguiente sección servirá para demostrar la existencia de variabilidad en el consumo energético al producir alimento concentrado, a continuación, se resumió en una tabla los consumos energéticos totales y las toneladas de alimento producido cada mes durante un año, luego se normalizó el consumo energético mensual llevándolo a kWh por tonelada producida.

| año mes | 2022 | | | | | 2023 | | | | | junio | julio |
|---------------|----------|------------|----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|
| | agosto | septiembre | octubre | noviembre | diciembre | enero | febrero | marzo | abril | mayo | | |
| kWh | 142632 | 139602 | 147840 | 163172 | 166967 | 173840 | 193447 | 191115 | 159760 | 153221 | 111818 | 120433 |
| ton producida | 6368,296 | 7325,905 | 5913,286 | 7864,796 | 7044,12 | 6562,972 | 8199,915 | 6717,948 | 7340,664 | 5796,423 | 5419,881 | 6241,865 |
| KPI kWh/tm | 22,40 | 19,06 | 25,00 | 20,75 | 23,70 | 26,49 | 23,59 | 28,45 | 21,76 | 26,43 | 20,63 | 19,29 |

Tabla 2: kWh/ton mensual

Luego, se graficaron estos consumos energéticos mensuales normalizados, resultando en el siguiente gráfico, donde se pueden apreciar que si existen variaciones mes a mes en los consumos energéticos normalizados en la producción de alimentos concentrados.

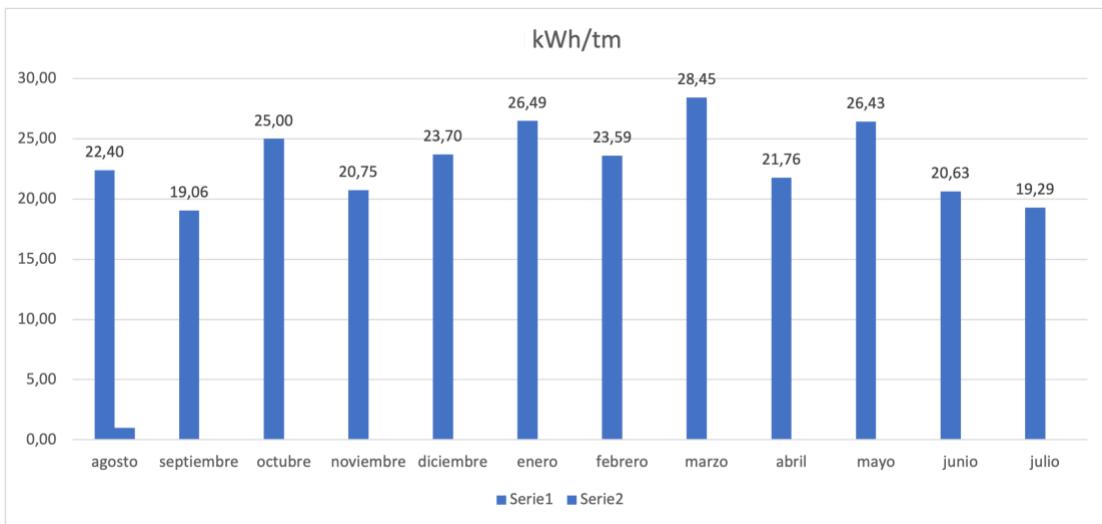


Ilustración 5: kWh/ton mensual

La necesidad de crear un modelo el cual sea capaz de predecir los consumos energéticos de las fórmulas de alimento concentrado que se quieran producir, lleva a un problema de regresión, para el cual debemos definir las variables dependientes como las independientes, las cuales las enlisto a continuación:

Variables independientes:

- Ingredientes:
 - Tricale, Maíz, Afrecho de trigo, Afrecho de soya, Urea, Afrecho de raps, Carbonato de calcio, Subproducto avena, Arvejas ingentec, Cáscara de arveja, Sales minerales.
- Formato:
 - Harinado
 - Peletizado (3, 6 o 8 mm)
- Cantidades:
 - Kilogramos

Variables dependientes:

- Consumo energético proceso molienda
- Consumo energético proceso peletizado

Objetivos

Objetivo general

Desarrollar un modelo que permita predecir los consumos energéticos en los procesos de mayor gasto energético en la producción del alimento concentrado, es decir, la molienda y el peletizado, al ingresar la fórmula que se quiera producir y el formato deseado, para saber los costos de producción y así definir qué tan rentable económicoamente es dicha fórmula.

Objetivos específicos

- Identificar y entender los distintos procesos y equipos de la línea de producción
- Identificar (y recopilar) qué datos hay disponibles para el desarrollo de del proyecto
- Analizar datos de consumos energéticos, datos de producción y datos de logística
- Desarrollar programa para el procesamiento de datos
- Desarrollar un programa para implementar, entrenar y hacer predicciones mediante un modelo de aprendizaje automático

Estado del arte

La búsqueda bibliográfica realizada con las palabras clave "planta de alimentos concentrados/balanceados", "predicción de consumo energético" y "machine learning" reveló una serie de documentos que abordan estas temáticas por separado, pero ninguno que integre específicamente el uso de machine learning para predecir consumos energéticos en una planta de alimentos concentrados. A continuación, se describen los estudios seleccionados y los aspectos relevantes identificados:

- **"Evaluación eléctrica y energética en la planta de alimentos balanceados para animales de la empresa AVUGA". Yimy David Vega Masís, (2020)**

Este estudio utilizó un equipo de medición (eGauge modelo 4115) para registrar variables eléctricas necesarias en el proyecto. Aunque centrado en la evaluación eléctrica, no aborda la aplicación de machine learning, ya que no se busca predecir consumos energéticos.

- **“APLICACIÓN DE TÉCNICAS DE MACHINE LEARNING PARA LA DETECCIÓN DE PATRONES DE CONSUMO ENERGÉTICO” Ruiz, Y. (2022).**

Este documento detalla cómo seleccionar y ajustar un modelo adecuado para la predicción de consumos energéticos, utilizando inteligencia artificial y machine learning. Sin embargo, se enfoca en la detección de patrones de consumo energético con series temporales, más que en su aplicación en la industria alimentaria.

En esta tesis se ocuparon distintas variaciones de modelos LSTM y modelos GRU, los cuales son útiles cuando existen series temporales en los datos, por lo tanto, no son útiles para este proyecto.

- **Evaluación energética en una planta de alimentos balanceado para animales. Costa Rica.”**
“Chacón, F. (2015).

En este proyecto se realiza una evaluación energética en una planta de alimentos balanceados de Costa Rica, con la finalidad de buscar oportunidades de maximizar el uso energético en sus operaciones. Esta empresa cuenta con la integración de un software y un analizador de distribución eléctrica para el monitoreo del consumo energético, específicamente disponen del software PowerNet y el analizador IQ Analyzer 6600 series, con lo cual buscan tener un control interno de sus consumos mensuales, por lo que el autor de este proyecto dispone de dicha información para llevar a cabo sus análisis, además, realiza mediciones con equipos manuales directamente en los procesos de estudio, en busca de oportunidades de mejora.

Dicha planta produce alrededor de 70 diferentes tipos de alimentos concentrados para bovinos, aves y cerdos. El autor se refiere a las variaciones que existen en el proceso de molienda debido a las distintas propiedades de la materia prima que se está procesando, pero a pesar de tener registro de datos de consumo eléctrico y reconocer estas variaciones de rendimiento en el proceso de molienda, para su análisis no las considera, simplemente se habla de “condiciones típicas de operación” cuando en la práctica están produciendo aproximadamente 70 fórmulas distintas. Por lo tanto, este estudio se enfoca en realizar una auditoría energética y encontrar oportunidades de mejora respecto a la

eficiencia energética y no a definir las relaciones existentes entre materia procesada y consumo energético como el presente proyecto.

- “Estado del Arte de la Predicción de Variables en Sistemas de Ingeniería Eléctrica Basada en Inteligencia Artificial”. Joseline Sánchez Solís, Marvin Coto Jiménez. (2021)

Este artículo científico analiza más de 300 artículos sobre la predicción de variables en sistemas de ingeniería eléctrica utilizando algoritmos de inteligencia artificial. Aunque los temas más citados se relacionan con la predicción de consumo eléctrico en edificaciones y energía solar, no se encontraron aplicaciones directas en la industria de alimentos concentrados.

“Los artículos más citados en la temática tratan sobre predicción de consumo eléctrico en edificaciones particulares, y de predicción de energía solar.” (Sánchez, Coto, 2021)

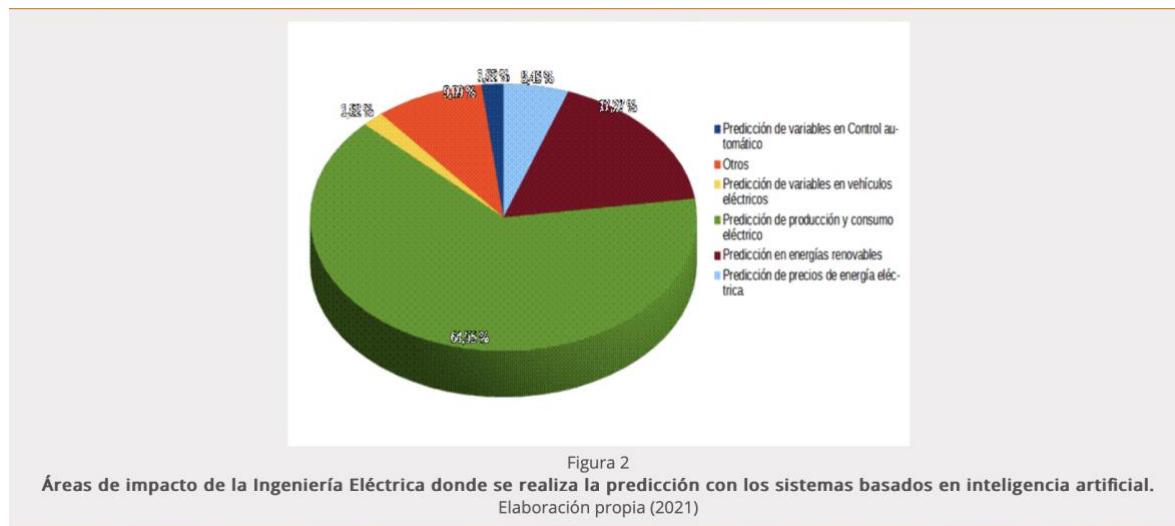


Ilustración 6: “Estado del Arte de la Predicción de Variables en Sistemas de Ingeniería Eléctrica Basada en Inteligencia Artificial” (Sánchez, Coto, 2021)

Se encontró que la herramienta más utilizada para predecir variables en sistemas de ingeniería eléctrica es el uso de redes neuronales, como se puede apreciar en el siguiente gráfico.

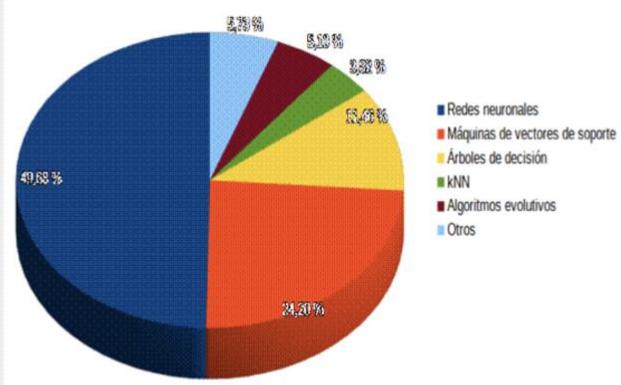


Figura 3
Porcentaje de las técnicas utilizadas en los artículos
Elaboración propia (2021)

Ilustración 7: "Estado del Arte de la Predicción de Variables en Sistemas de Ingeniería Eléctrica Basada en Inteligencia Artificial" (Sánchez, Coto, 2021)

Estos estudios proporcionan un contexto útil, pero también resaltan un vacío en la investigación específica sobre el uso de machine learning para predecir el consumo energético en la industria de alimentos concentrados. Este vacío representa una oportunidad única para este proyecto, donde se busca llenar esta brecha mediante la aplicación de técnicas de machine learning avanzadas para mejorar la gestión de costos en los procesos de producción.

Soluciones propuestas

Para abordar un problema de regresión como lo es predecir el consumo energético de un proceso basado en las materias que se están procesando, existen múltiples alternativas. La elección del método adecuado dependerá de la naturaleza del problema y de los datos, de la complejidad del proceso y la precisión requerida en las predicciones. Dado el problema específico de predecir el consumo energético en los procesos de molienda y peletizado, se consideraron varias soluciones comparables.

Las siguientes son tres opciones viables:

1. Modelo de Regresión Lineal Múltiple:

- **Descripción:** Este modelo establece una relación lineal entre las variables independientes (ingredientes y formatos de producción) y la variable dependiente (consumo energético). Es sencillo de implementar y entender.
- **Ventajas:** Fácil interpretación de los resultados y bajo costo computacional.

- **Desventajas:** Puede no capturar relaciones complejas o no lineales entre las variables.

2. Random Forest Regressor:

- **Descripción:** Es un método de ensamble que utiliza múltiples árboles de decisión para realizar predicciones más precisas y robustas.
- **Ventajas:** Mejor capacidad para modelar relaciones no lineales y complejas, proporciona una medida de la importancia de las características.
- **Desventajas:** Más complejo y computacionalmente costoso que la regresión lineal, puede ser más difícil de interpretar.

3. Red Neuronal Artificial:

- **Descripción:** Una red neuronal multicapa (MLP) que puede capturar relaciones complejas y no lineales en los datos.
- **Ventajas:** Alta flexibilidad y capacidad para modelar interacciones complejas entre variables.
- **Desventajas:** Requiere una mayor cantidad de datos para entrenamiento efectivo, es computacionalmente intensivo y puede ser difícil de interpretar.

Solución a implementar

La solución escogida para lograr encontrar las relaciones entre las materias primas y su consumo energético en los procesos de producción y poder predecir correctamente estos consumos es un modelo de aprendizaje automático.

Se graficaron las cantidades de los ingredientes vs la energía consumida para cada producción y para ambos procesos, descubriéndose que no existe una relación lineal, por lo que el modelo a usar debe ser capaz de captar relaciones complejas entre las variables.

Se probaron los siguientes modelos para comparar su rendimiento y seleccionar el más adecuado a los datos de entrenamiento:

- `MLPRegressor`
- `SVR`
- `RandomForestRegressor`

- GradientBoostingRegressor
- AdaBoostRegressor
- GradientBoostingRegressor
- HistGradientBoostingRegressor
- XGBRegressor
- CatBoostRegressor
- LGBMRegressor

Después de evaluar las opciones, se decidió implementar el MLPRegressor de Scikit-learn para este proyecto. Esta elección se basó en la naturaleza compleja del problema y la necesidad de capturar relaciones no lineales entre las variables de entrada y el consumo energético. El MLPRegressor, con su capacidad para modelar estas complejidades y su flexibilidad en la arquitectura de la red, se consideró la opción más adecuada. A continuación, se muestra una tabla (Tabla 3) con los modelos evaluados y su correspondiente coeficiente de determinación (R2):

| Modelo | R2 |
|-------------------------------|------|
| MLPRegressor | 0,82 |
| HistGradientBoostingRegressor | 0,71 |
| RandomForestRegressor | 0,74 |
| GradientBoostingRegressor | 0,70 |
| XGBRegressor | 0,70 |
| LGBMRegressor | 0,68 |
| SVR | 0,75 |
| AdaBoostRegressor | 0,76 |
| CatBoostRegressor | 0,79 |

Tabla 3: Modelos evaluados.

Se desarrollaron los siguientes programas para llevar a cabo este proyecto, el primero corresponde el programa de procesamiento de datos, luego está el programa usado para reformatear los datos extraídos del SCADA, luego el código para concatenar los archivos de entrenamiento semanales, el siguiente código es donde se normalizan y codifican los datos de entrenamiento y se implementan, entranan y evalúan diferentes modelos predictivos, además se busca el mejor ajuste de hiperparametros, usando funciones como RandomizedSearchCV y GridSearchCV, finalmente se desarrolló otro programa para realizar análisis de estos datos y encontrar algunas conclusiones útiles

para la empresa. Los códigos de estos programas están disponibles en el Anexo y los principales programas son explicados en detalle en el capítulo de Metodologías.

Riesgos y sus mitigaciones

Este es un proyecto con baja probabilidad de riesgos, ya que se trata de la implementación de modelo predictivo, pero de igual manera los posibles riesgos son:

- Un bajo rendimiento/precisión de modelo predictivo debido a la elección incorrecta del modelo.
- Un mal uso de la herramienta de predicción o mal rendimiento del modelo.
- Problemas con la lectura de los datos del SCADA debido a su formato

Y sus mitigaciones son las siguientes:

- Se desarrolló un programa que evalúa diferentes modelos, comparando sus métricas de precisión y rendimiento, como el error cuadrático medio, para elegir el modelo que más se ajusta a los datos de entrenamiento.
- Probar nuevos ajustes de hiperparametros.
- Nuevos criterios de filtrado/mejorar calidad de datos.
- Capacitación del personal de la empresa que usara la herramienta de predicción, para que puedan seguir entrenando el modelo con nuevos datos y mantener un rendimiento optimo del mismo.
- Se desarrolló un pequeño código para reformatear los datos del SCADA y que el programa los pueda interpretar correctamente, este problema se debía a que el separador, como el decimal de los valores era una coma, por lo que los datos se mezclaban y se perdía la lógica de estos.

Evaluación económica

Este proyecto no tiene un gasto económico asociado, por lo que en esta sección se analizará el potencial impacto que puede generar esta iniciativa en la empresa.

La implementación de un modelo predictivo que sea capaz de anticipar el consumo energético y, por lo tanto, el costo económico asociado a la producción de un alimento específico, busca eliminar la incógnita que existe actualmente en el proceso productivo y proporcionar información valiosa a los tomadores de decisiones comerciales de la empresa. Por ejemplo, permitirá diseñar estrategias de ventas basadas en los costos totales, que incluyen tanto la materia prima como la energía utilizada en la producción, de los diferentes alimentos ofrecidos.

Metodologías

1. Identificar y entender los distintos procesos y equipos de la línea de producción

Se realizó un reconocimiento en terreno del proceso productivo y de los equipos, además de un estudio de motores, así como sus potencias y sus variaciones de carga.

2. Identificar que datos hay disponibles para el desarrollo del proyecto

Se realizó una búsqueda de datos en 3 bases de datos disponibles en la empresa, estas son: software SCADA, base de datos área logística y base de datos área producción.

Se observó que en las 3 bases de datos existían datos de alto valor que podían ser utilizados para el desarrollo de este proyecto.

3. Análisis de datos de consumos energéticos, datos de producción y datos de logística

Recopilar datos de consumo de energía en los procesos de molienda y peletizado entregados por el SCADA, datos de producción que contienen las fechas y horas de cada producción y datos de logística que contienen los ingredientes, formato y cantidad de cada producción.

4. Desarrollo de programa para el procesamiento de datos

Se desarrolló un programa en la plataforma jupyter, utilizando el lenguaje Python y las bibliotecas pandas y numpy, con la finalidad de que procese, ordene, limpie, integre, normalice y alineé temporalmente las distintas bases de datos que sean entregadas a este y entregue un documento unificado con la información y las variables necesarias para entrenar un modelo de aprendizaje automático.

Este programa fue utilizado semana a semana para procesar y crear los dataframe con los datos de entrenamiento, esto debido a que la memoria del software SCADA de la empresa únicamente guarda las lecturas de amperaje durante 7 días, debido a esta limitación, fue necesario ir a la sala de operadores de la planta cada lunes para exportar el registro de los datos manualmente de los últimos 7 días, además de esos datos

5. Desarrollo de un programa para implementar, entrenar y hacer predicciones mediante un modelo de aprendizaje automático

Desarrollar un programa para la implementación de un modelo de aprendizaje automático en la plataforma jupyter, utilizando el lenguaje Python y la biblioteca scikit learn, en el cual ingrese los datos unificados por el programa anterior para entrenarlo y realizar predicciones de los consumos energéticos de distintas fórmulas de alimento concentrado.

6. Entrenar y validar modelo

Entrenar y evaluar diferentes modelos mediante métricas de precisión y desempeño como; MAE, MSE, ERR_MAX, R2. Seleccionar el modelo con mejor rendimiento para el conjunto de datos de entrenamiento y ajustar sus hiperparametros para modificar la arquitectura, la tasa de aprendizaje y la complejidad del modelo, buscando maximizar su precisión de predicción.

Medidas de desempeño

Las medidas de desempeño a usar para evaluar el éxito de este proyecto serán el coeficiente de determinación (R2) (Ecuación 1), el error cuadrático medio (MSE) (Ecuación 2) , el error absoluto medio (MAE) (Ecuación 3) del modelo predictivo, usando datos del consumo energético real en el

proceso de molienda y en el proceso de peletizado y el consumo energético estimado por el modelo utilizado, para esto utilizaré un conjunto de datos de validación, es decir, datos que no hayan sido usados para entrenar el modelo previamente. A continuación se presentan las fórmulas matemáticas de las métricas utilizadas:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Ecuación 1

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

Ecuación 2

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

Ecuación 3

Donde:

- y representa los valores reales del consumo energético.
- \hat{y} representa los valores predichos del consumo energético.
- \bar{y} representa el valor medio del consumo energético.

Desarrollo del proyecto basado en la metodología e implementación

A continuación, se explicará en detalle cómo funciona el programa desarrollado para el procesamiento de datos. Las entradas a este programa son cinco archivos, dos de estos son las lecturas de amperaje del molino 1 y del molino 2, estos archivos contienen distintas variables del proceso, de las cuales nos interesan solo dos: la primera es el tiempo del dato (Actual Current Time) y la segunda es el valor del dato (Actual Current ValueY). Luego, un archivo de la base de datos de logística, el cual contiene las órdenes de venta y todo tipo de detalles de estas, como sus ingredientes, sus análisis de laboratorio

e información sensible, como datos del vendedor, del cliente, del transportista, precio, información de facturación, etc. En total son 451 columnas de datos para cada producción, de las cuales se hace una limpieza y solo se deja la información relevante para el programa. En este caso, se buscan los datos de ingredientes para cada producción. Finalmente, se ingresan al programa dos archivos más, los cuales son las planillas de producción, una para cada línea. Estas planillas en formato Excel contienen el registro de los tiempos de inicio y término de cada producción en los distintos procesos de la línea productiva, además de su ID, nombre de la fórmula, cliente y otros datos que no son relevantes para el análisis. Una vez cargados estos archivos y seleccionados los datos que serán utilizados en el programa, se procede a unir en un único documento las órdenes y las planillas de producción. Esto se realiza mediante la función merge, la cual une documentos según una variable en común, en este caso el ID, para no perder la alineación de los datos. Luego, se hace una sumatoria de todas las sales presentes en las órdenes, ya que existen más de 300 sales distintas en el inventario, por lo que se reemplazan todas esas columnas por una sola llamada 'suma de sales' que contiene el total de estas. A continuación, se utiliza una función creada específicamente para calcular la energía usada por cada producción en el proceso de molienda. Esta función lee las horas de inicio y término, que previamente fueron unidas a la fecha y transformadas a formato datetime, con el propósito de poder compararlas con las fechas de los archivos que contienen las lecturas de amperaje. Una vez identificado el inicio y el término del proceso de molienda, la función procede a leer los valores del amperaje segundo a segundo entre el intervalo seleccionado y calcular su promedio. Ese valor es ingresado a una fórmula (Ecuación 4) la cual calcula la energía en kWh, utilizando el voltaje del equipo, el factor de planta y la raíz de 3 (corriente trifásica).

$$\text{Energía (kWh)} = \text{promedio amperaje (1 seg)} * \text{tiempo intervalo (seg)} * \frac{1}{3600} * \sqrt{3} * 390 * 0.89 / 1000$$

Ecuación 4

Mediante un ciclo “for”, la función calcula la energía para cada una de las producciones e ingresa ese valor en una nueva columna llamada ‘energía molino’. Finalmente, el archivo se exporta en formato CSV con el nombre de “datos entrenamiento HM” (HM= hammer mill).

Este mismo proceso se replica con los archivos del proceso de peletizado, resultando en el archivo “datos entrenamiento PP”. A continuación, se presenta un diagrama de flujo simplificado del programa recién descrito.

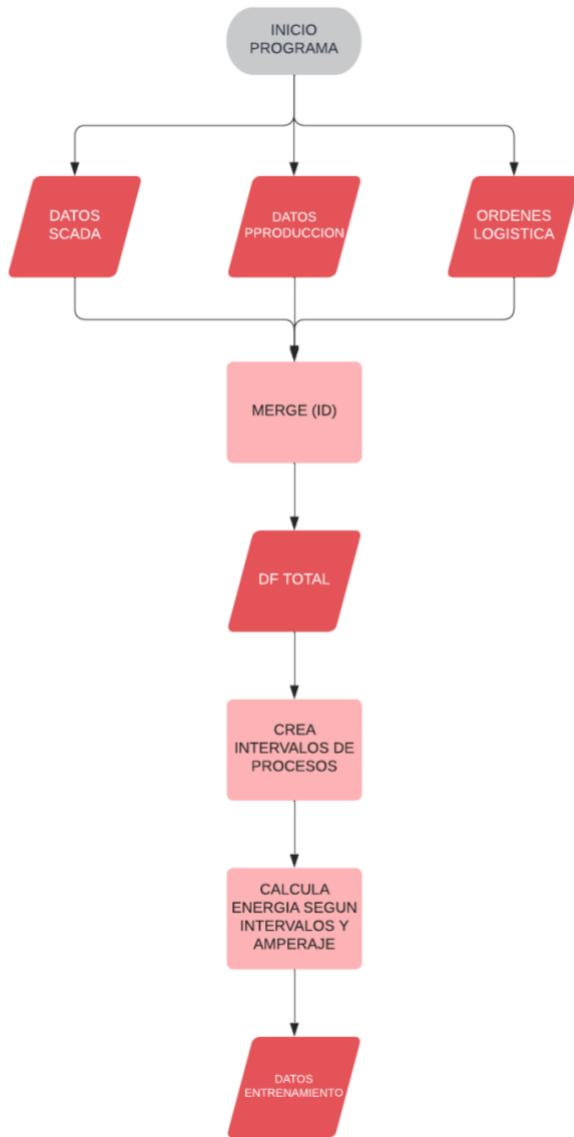


Ilustración 8: diagrama de flujo programa procesamiento de datos (elaboración propia)

Además, se desarrolló el programa para la selección y entrenamiento del modelo de aprendizaje automático, en el cual se entrenaron 10 modelos con el conjunto final de datos de entrenamiento, previamente escalado con la función MinMaxScaler y codificado con la función OneHotEncoder (variables categóricas), el cual está formado por 14 semanas de datos para el proceso de molienda y 8 semanas para el proceso de peletizado, luego se compararon los rendimientos de los modelos mediante las siguientes métricas; R2_score, MSE, MAE y max_error.

Resultados cualitativos y cuantitativos

Resultados cuantitativos:

El modelo que logró los mejores rendimientos con el conjunto de datos y el que será utilizado, es el MLPRegressor de scikit-learn, este una red neuronal de perceptrón multicapa diseñado específicamente para problemas de regresión, el modelo dio un r2_score de 0.82 y un MAE de 9.71 para la predicción de consumo del proceso de molienda y un r2_score de 0.71 y un MAE de 16.64 para la predicción de consumo del proceso de peletizado.

Mediante el análisis de los datos utilizados para entrenar los modelos, se destacaron varios hallazgos significativos relacionados con el consumo de energía en los procesos de molienda y peletizado. Se observó que la Línea 2 consume un 38% más de energía, en la molienda como en el peletizado, que la Línea 1. En el proceso de peletizado, se encontraron diferencias notables en el consumo de energía según el tamaño del pellet: peletizar en 6mm consume un 14.5% más de energía que en 8mm para ambas líneas, y peletizar en 3mm requiere un 25.4% más de energía que en 8mm. Además, al peletizar en 3mm, la Línea 2 consume un 31% más de energía que la Línea 1, en 6mm la Línea 2 consume un 62% más de energía, y en 8mm un 24% más de energía que la Línea 1.

Resultados cualitativos:

Los resultados cualitativos más significativos de la implementación del modelo predictivo en Alisur es la mejora en la toma de decisiones estratégicas y comerciales. Este modelo elimina la incertidumbre previamente asociada con los costos energéticos del proceso de molienda y peletizado, aspectos cruciales en la producción de los distintos alimentos. Al proporcionar predicciones sobre el consumo energético, el modelo permite a Alisur planificar y asignar recursos con mayor eficiencia, optimizando así tanto el rendimiento como los costos de producción. Esta herramienta se convierte en un elemento esencial para la empresa, mejorando su capacidad para tomar decisiones comerciales informadas y estratégicas.

Conclusiones y discusión

En el análisis de los resultados del proyecto en Alisur, se identificó que las interrupciones en los procesos de molienda y peletizado pueden ser un factor clave en el rendimiento subóptimo de los modelos predictivos. Estas detenciones, debidas a una variedad de razones operativas, tienen el

potencial de alterar las estimaciones de consumo energético, lo que afecta la precisión de las predicciones de los modelos.

Además, es relevante señalar que parte de los datos utilizados para calcular el consumo de energía proviene directamente de los operadores. Aunque estos datos son cruciales para el funcionamiento del modelo, su naturaleza subjetiva y la posibilidad de errores humanos o inconsistencias en la recopilación pueden comprometer la exactitud de la información. Esta variabilidad en la calidad de los datos puede impactar significativamente en la precisión de los modelos predictivos.

Por lo tanto, se recomienda un ajuste continuo del modelo, entrenándolo con nuevos datos y considerando estas posibles fuentes de error. A pesar de que el modelo predictivo desarrollado para Alisur es una herramienta valiosa para la toma de decisiones, su efectividad está estrechamente ligada a la calidad y consistencia de los datos de entrada, así como a la estabilidad de los procesos productivos. La mejora continua en estas áreas será fundamental para maximizar el potencial del modelo.

Las diferencias encontradas entre la Línea 1 y la Línea 2 pueden explicarse por la disparidad en la optimización y programación de las líneas, ya que la Línea 2 tiene poco más de un año en operación, mientras que la Línea 1 lleva casi 10 años en funcionamiento. Esta distinción en la experiencia operativa podría influir en la eficiencia energética de cada línea.

Referencias

- **Sánchez Solís, J. & Coto Jiménez, M. (2021). Estado del Arte de la Predicción de Variables en Sistemas de Ingeniería Eléctrica Basada en Inteligencia Artificial. Universidad de Costa Rica.**
- **Ruiz, Y. (2022). Aplicación de técnicas de machine learning para la detección de patrones de consumo energético. Universidad de Málaga.**
- **“Chacón, F. (2015). Evaluación energética en una planta de alimentos balanceado para animales. Universidad de Costa Rica.”**

- Vega-Masís, Y. & Guerrero-Castro, O. (2020). Evaluación eléctrica y energética en la planta de alimentos balanceados para animales de la empresa AVUGA. Instituto Tecnológico de Costa Rica.

Anexo

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

In [ ]: #importar df corriente HM
df_HM01 = pd.read_csv("HM01_30.04.csv", encoding="utf8")
df_HM02 = pd.read_csv("HM02_modified.csv", encoding="utf8")
#borrar otras variables
df_HM01 = df_HM01.drop(["Feeder Speed Time", "Feeder Speed ValueY"], axis=1)
#df_HM02 = df_HM02.drop(["Feeder Speed Time", "Feeder Speed ValueY", "Setpoint"])
#cambiar formato fecha
df_HM01["Actual Current Time"] = pd.to_datetime(df_HM01["Actual Current Time"])
df_HM02["Actual Current Time"] = pd.to_datetime(df_HM02["Actual Current Time"])
#lee una fila por medio y resetear indice si los datos estan duplicados
if df_HM01.iloc[0]["Actual Current Time"] == df_HM01.iloc[1]["Actual Current Time"]:
    # Restablecer el índice solo si el primer valor se repite
    df_HM01 = df_HM01[::2].reset_index(drop=True)

In [ ]: ## importa ordenes logistica
ordenes = pd.read_csv("orders 30-04.csv", sep=';', encoding="utf8")
pd.set_option('display.max_columns', None)

In [ ]: #importa datos produccion linea 1 y 2
linea1 = pd.read_excel("linea1-30.04 nov.xlsx")
linea2 = pd.read_excel("linea2-30.04 nov.xlsx")
df_produccion = pd.concat([linea1, linea2], axis=0, join="inner", ignore_index=True)
pd.set_option('display.max_rows', None)

In [ ]: ## unir df logistica y df produccion segun ID en comun
df_total = pd.merge(df_produccion, ordenes, on="ID", how='inner')

In [ ]: # elimina columnas especificadas
df_total = df_total.drop(["Nombre de Fórmula", "KG", "Produccion", "FORM", "",
                           "Tipo de venta", "Estado", "Pago", "Fecha Orden", "",
                           "Rut Empresa Cliente", "Vendedor", "Nombre ubicaci",
                           "Fecha Término", "Fecha término producción", "",
                           "Fecha Despacho", "Fecha Entrega", "¿Servicio de c",
                           "Proteína (%)", "Materia grasa (%)", "Pradera (kg/",
                           "Forraje III (kg/día)", "Pc", "Em", "Fdn", "Fda", "",
                           "Valor estimado (con iva)", "Precio por kg", "Valo",
                           "Entrega urgente (sin iva)", "Descarga de sacos (s",
                           "Servicio de venta (sin iva)", "Método de Pago", "",
                           "Tipo de carrocería", "Nombre Chofer", "Rut Chofer", "",
                           "Fecha llegada de camión", "Responsable producción", "",
                           "Fecha primer pesaje", "Responsable despacho", "F",
                           "Empresa que emite la guía/factura", "Número de Gu",
                           "Presencia de insectos", "Confirmada", "Fecha últi",
                           "Penúltimo producto transportado", "Ante penúltim",
                           "¿Hay residuos del último producto transportado?", "",
                           "Temperatura (°C)", "Peso Hectolitro (Kg/hl)", "H",
                           "Humedad lenta 3 (%)", "Humedad rápida (%)", "Hume",
                           "Peso muestra limpia (g)", "Impurezas/Finos (%)", "",
                           "Granos partidos (%)", "Peso granos germinados (g)"])
```

```

    "Granos chupados (%)", "OP", "Cliente", "TOLVA", "Tiempo molienda (minutos)", "Fecha inicio dosific
    "Velocidad de producción (ton/hr)", "Velocidad alim
    "Temperatura acondicionador (°C)", "Velocidad alimentación", "Dureza pellet (kg/cm2)", "Capacity Setpoint feed
    "Setpoint Temperatura Prensa (C)", "Inicio Dosificación"
# Identifica todas las columnas que son NaN
nan_cols = df_total.columns[pd.isna(df_total.columns)]
```

```

# Elimina esas columnas
df_total = df_total.drop(nan_cols, axis=1)
```

```
In [ ]: # Eliminar filas donde cualquier celda tenga un valor NaN
df_total = df_total.dropna()

# Restablecer el índice del DataFrame
df_total = df_total.reset_index(drop=True)
```

```
In [ ]: #unir columna fecha a columna inicio molienda y a termino molienda y cambia formato
df_total["Inicio Molienda"] = pd.to_datetime(df_total["fecha"].astype(str) + " 00:00:00")
df_total["Termino Molienda"] = pd.to_datetime(df_total["fecha"].astype(str) + " 23:59:59")
#eliminar columna fecha
df_total = df_total.drop("fecha",axis=1)
```

```
In [ ]: #suma columnas sales, crea columna total sales y borra las sales, cambia datos
sales = ["Px crucero viejo plus", "Alisur 7000+ (sin premix)", "Alisur 5000", "Mix base 8(borrar)", "Mix elemental 8(borrar)", "Mix pastoreo 7(borr
    "Mix elemental 7(borrar)", "Mix base 6(borrar)", "Base podológica 6(borr
    "Mix base podológica", "Mix crianza 3(borrar)", "Mix crianza 5(borr
    "Mix elemental 2(borrar)", "Mix pastoreo 2(borrar)", "Mix crianza 1(borr
    "Mix crianza 2(borrar)", "Mix crianza plus", "Mix crianza plus 1(borr
    "Sal mineral eh (borrar)", "Sal mineral eh", "Meggi 5 (borrar)", "M
    "Mipro close up (borrar)", "Mix elemental podológica", "Alisur podolog
    "Mipro close up 350", "Meggi 5", "Meggi 5 r", "Mipro 400", "Mipro h
    "Alisur podológica bcs (borrar)", "Alisur gp podológica bcs", "Ali
    "Action 100 b", "Action pradera se", "Pmx a100 bb", "Action 100", "P
    "Premix 17 medicado", "Premix 30", "Premix 35 medicado", "Premix 7", "P
    "Premix hualleria 1 (borrar)", "Premix 9 ec", "Podologico medicada"
    "Premix 17 medicado (borrar)", "Premix 35 medicado (borrar)", "Mix
    "Premix alisur pb200u (borrar)", "Premix 6", "Premix hualleria ", "P
    "Premix alisur 7200u (borrar)", "Premix alisur 105u (borrar)", "Meg
    "Premix alisur 5165u (borrar)", "Premix crucero viejo 8614u (borra
    "Reproceso", "Alisur gp 7000+ (borrar)", "Alisur gp 7000+", "Premix
    "Premix 5", "Premix action 100 bbu (borrar)", "Premix 9 ec (borrar)
    "Mix crucero viejo plus (borrar)", "Núcleo crianza alisur", "Alisur
    "Premix 30 (borrar)", "Nutrasal integral", "Intelisal jersey", "Meg
    "Premix 10", "Mipro close up r 350", "Mipro 350 macro se", "Mipro p
    "Mineral forte", "Gekl josera", "Alisur pastoreo (sin premix)", "Nu
    "Mix elemental podol.-coop", "Mipromast r 200 feedlot", "Premix 23", "M
    "Mipro hp l 400", "Intelisal coagra acid buf", "Intelisal pastoreo"
    "Megafat 88", "Mega ade se rumensin", "Mipro hp 250 5p", "Nutrialmi
    "Mix reposición plus", "Salfort alta rumia", "Mipro 250 nu", "Salf
    "Sal mineral la carmela", "Nutrialmix aniónica", "Min. forte pod. p
    "Nutrialmix preparto", "Premix antillanca", "Mipro spring 200", "Óx
    "Anamix preparto 100", "Anamix cabra alta full dr. cesped", "Anamix
```

```

"Alisur preparo aniónico", "Premix antillanca plus", "Premix cromo
"Sal grassland", "Premix 29 medicado", "Acid buf", "Vetersal seleni
"Fe manuka lechería rumensin", "Intelisal pastoreo acidbuf", "Mix t
"Vetersal alta produccion", "Premix 40 medicado", "Bicarbonato de s
"Nutrialmix futrono a buf", "Nitrofeed", "Fe manuka pastoreo", "Mix
"Biocolina", "Pradera primavera muy buena", "Pradera primavera buer
"Pradera verano muy buena", "Pradera verano buena", "Pradera veranc
"Pradera otoño buena", "Pradera otoño regular", "Pradera invierno n
"Pradera invierno regular", "Silo pradera muy bueno", "Silo pradera
"Silo maíz bueno", "Silo maíz regular", "Vetersal me preparo full"
"Vetersal lactancia manuka", "Vetersal full sól. c/rumensin", "Prem
"Nutrialmix pastoreo biose", "Lechería estándar", "Mipromast r feed
"Premix 48 medicado", "Meganion lc", "Rmv manuka pastoreo rumensin"
"Mipro lechería r 200", "Mipro hp 250 btn", "Mix vaca preparto", "J
"Anamix dairy pack", "Nutrialmix manuka lechería", "Alisur gp 7000
"Mix lactancia puduhue plus", "Mipro pastoreo skyline", "Oxi alisur
"Mipro hp 400", "Nutrialmix pastoreo plus", "Mix preparto clor. col
"Nutrasal Óptima", "Nucleo similk ternero starter", "Premix los pel
"Nutrialmix pasture feed", "Anamix integra full lactancia rumensin"
"Fe ferosor biotina", "Safetox plus", "Mipro pastoreo r skyline pro
"Mineral lactancia treimun plus", "Mix los copihues", "Micofix plus
"Nutrasal free stall", "Lactancia futahuente", "Premix 7000 alisur"
"Fe sólidos full r 200", "Premix 5000 alisur", "Premix 40 med", "Px
"Alisur pastoreo", "Vetersal sólidos full", "Mix base pod. plus", "
"Mix preparto aniónica chlor", "Mineral l- treimun millacura plus",
"Vetersal cardal lactancia c/rumensin", "Vetersal la cabaña lactanc
"I. pastoreo acid buf", "Tres esteros", "Mipro lechería btn 200", "
"Mineral forte buf", "Intelisal lactancia", "Vetersal r.m.v sal mir
"Mineral mega podológica", "Nutrialmix podal sur", "Preparto treim
"Mix preparto puduhue aac plus", "Salfort lechera estándar", "Intel
"Mipromast feedlot", "Mix preparto aniónico especial", "Megamix mg
"Intelisal pastoreo purranque", "Nutrialpack anionplus", "Fdo. sn.
"Preparto grand cru lisina", "Super organica grand cru", "Biotina c
"Buffin grand cru", "Mineral mega ade org", "Mineral lactancia futa
"Mipro spring r 200", "Vetersal lecheria basica", "Megamix crianza
def replace_comma(val):
    if pd.isna(val):
        return val
    val = str(val).replace(',', '.')
    try:
        return float(val)
    except ValueError:
        return val
df_total[sales] = df_total[sales].map(replace_comma)
df_total['Suma Sales'] = df_total[sales].sum(axis=1)
df_total = df_total.drop(columns=sales, axis=1)

```

```

In [ ]: def consulta_df_avg(df1, df2, inicio, fin, parametro):
    # Determina qué DataFrame usar basado en el parámetro
    df_usar = df1 if parametro == 0 else df2

    # Convertir las fechas de inicio y fin a objetos datetime de pandas
    inicio = pd.to_datetime(inicio)
    fin = pd.to_datetime(fin)

    # Crea máscaras booleanas para filtrar el DataFrame entre las fechas de

```

```

mask = (df_usar["Actual Current Time"] >= inicio) & (df_usar["Actual Cur
df_filtrado = df_usar.loc[mask]

# Inicializa el valor del promedio a 0
avg_value = 0

# Si el DataFrame filtrado no está vacío, calcula el promedio ponderado
if not df_filtrado.empty:
    avg_value = df_filtrado['Actual Current ValueY'].mean()
    interval_time = (fin - inicio).total_seconds() / 3600
    avg_value = avg_value * interval_time * np.sqrt(3) * 390 * 0.89 / 10

return df_filtrado, avg_value

# Supongamos que df_total está definido y tiene las columnas "Inicio Molienda" y "Termino Molienda"

# Inicializa las listas para almacenar los resultados
resultados = []
promedios = []

# Bucle que recorre df_total
for i in range(len(df_total)):
    inicio = df_total.loc[i, "Inicio Molienda"]
    fin = df_total.loc[i, "Termino Molienda"]
    parametro = df_total.loc[i, "Planta Proceso"]

    # Obtiene el DataFrame filtrado y el valor del promedio
    resultado, promedio = consulta_df_avg(df_HM01, df_HM02, inicio, fin, parametro)

    # Añade el resultado y el valor del promedio a las listas
    resultados.append(resultado)
    promedios.append(promedio)

# Agrega los resultados integrales a nueva columna en df_total
df_total["energia molino"] = promedios

```

```

In [ ]: # Eliminar filas donde cualquier celda tenga un valor NaN
df_total = df_total.dropna()
# Eliminar filas donde la energia es 10 kWh
df_total = df_total[df_total["energia molino"] > 10]
# Restablecer el índice del DataFrame
df_total = df_total.reset_index(drop=True)

```

```

In [ ]: #exporta df a csv
df_total.to_csv("entrenamiento 30.04 nov.csv", index=False)

```

```

In [ ]:

```

```
In [1]: import pandas as pd
import re

In [2]: def reformat():
    with open("PP02_30.04.csv", encoding="utf-8") as file:
        lines = file.readlines()

        # Tomar el tercer y cuarto elemento del encabezado
        head = lines[0].split(',') [2:4]
        new_lines = []
        for line in lines[1:]:
            # Buscar todas las ocurrencias de fechas y sus valores asociados en
            pattern = r'(\d{2}/\d{2}/\d{4} \d{1,2}:\d{2}:\d{2}),(\d+\.\?\d*)'
            matches = re.findall(pattern, line)

            if len(matches) >= 2:
                # Tomar la segunda fecha y el valor que le sigue
                second_date, value_after_second_date = matches[1]
                new_line = f'{second_date},{value_after_second_date}'
                new_lines.append(new_line)
            else:
                # Manejar el caso de que no se encuentren suficientes fechas
                print(f"No se encontraron suficientes fechas en la línea: {line}")

        with open('PP02_modified.csv', 'w', encoding='utf8') as file:
            file.write(','.join(head) + '\n')
            for line in new_lines:
                file.write(line + '\n')

reformat()
```

In []:

```
In [ ]: import pandas as pd
```

```
In [ ]: df1 = pd.read_csv("23-27 agosto/entrenamiento 23.27 agosto.csv")
df2 = pd.read_csv("28-02 sept/entrenamiento 28.02 sept.csv")
df3 = pd.read_csv("04-09 sept/entrenamiento 04.09 sept.csv")
df4 = pd.read_csv("11-14 sept/entrenamiento 11.14 sept.csv")
df5 = pd.read_csv("15-16 sept/entrenamiento 15.16 sept.csv")
df6 = pd.read_csv("20-23 sept/entrenamiento 20.23 sept.csv")
df7 = pd.read_csv("25-30 sept/entrenamiento 25.30 sept.csv")
df8 = pd.read_csv("03-07 oct/entrenamiento 03.07 oct.csv")
df9 = pd.read_csv("10-14 oct/entrenamiento 10.14 oct.csv")
df10 = pd.read_csv("16-21 oct/entrenamiento 16.21 oct.csv")
df11 = pd.read_csv("23-26 oct/entrenamiento 23.26 oct.csv")
df12 = pd.read_csv("30-04 nov/entrenamiento 30.04 nov.csv")
df13 = pd.read_csv("06-11 nov/entrenamiento 06.11 nov.csv")
df14 = pd.read_csv("14-18 nov/entrenamiento 14.18 nov.csv")
```

```
In [ ]: df_training = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9, df10,
```

```
In [ ]: # Eliminar filas donde la energía es 10 kWh
df_training = df_training[df_training["Peso Total (Kg)"] != 0]

# Identifica los valores mínimos y máximos de la columna objetivo
min_value = df_training['energia molino'].min()
max_value = df_training['energia molino'].max()

# Elimina las filas con valores mínimos y máximos
df_training = df_training[df_training['energia molino'] != min_value]
df_training = df_training[df_training['energia molino'] != max_value]
print(max_value, min_value)
```

```
In [ ]: # Eliminar columnas
df_training = df_training.drop(["Termino Prensa / Termino Harinado", "Inicio",
'Raps oleotop', 'Trigo nutreco', 'Cebada', 'Col forrajera', 'Nabo forrajero', 'Raps forraje', 'Cáscara de soya', 'Harina de soya (borrar)'])
```

```
In [ ]: #exporta df a csv
df_training.to_csv("datos entrenamiento HM 8W.csv", index=False)
```

```
In [ ]:
```

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import max_error
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
import pickle
```

```
In [ ]: # Cargar tus datos en un DataFrame de Pandas
df = pd.read_csv("datos_entrenamiento HM RAW.csv", sep=',', encoding="utf8")
```

```
In [ ]: # Codificar con OneHotEncoder
encoder_planta_proceso = OneHotEncoder()
encoder_formato = OneHotEncoder()

# Codificar la columna "Planta Proceso"
encoded_planta_proceso = encoder_planta_proceso.fit_transform(df[["Planta Proceso"]]).toarray()

# Codificar la columna "Formato"
encoded_formato = encoder_formato.fit_transform(df[["Formato"]]).toarray()

# Crear DataFrames a partir de los arrays codificados
df_encoded_planta_proceso = pd.DataFrame(encoded_planta_proceso, columns=encoder_planta_proceso.get_feature_names_out())
df_encoded_formato = pd.DataFrame(encoded_formato, columns=encoder_formato.get_feature_names_out())

# Unir los DataFrames codificados con el DataFrame original
# Asegúrate de restablecer el índice si es necesario para evitar problemas con las fechas
df.reset_index(drop=True, inplace=True)
df_encoded_planta_proceso.reset_index(drop=True, inplace=True)
df_encoded_formato.reset_index(drop=True, inplace=True)

df = pd.concat([df, df_encoded_planta_proceso, df_encoded_formato], axis=1)

# Opcionalmente, eliminar las columnas originales si ya no son necesarias
df.drop(["Planta Proceso", "Formato"], axis=1, inplace=True)
```

```
In [ ]: # Normalizar con MinMaxScaler
```

```
scaler = MinMaxScaler()

norm_col = ['Peso Total (Kg)', 'Triticale', 'Maíz', 'Afrecho de trigo', 'Afr
            'Carbonato de calcio', 'Subproducto avena', 'Arvejas ingentec',
            ]
for col in norm_col:
    if df[col].dtype in ["object", "str"]:
        df[col] = df[col].str.replace(',', '.').astype(float)
    else:
        df[col] = df[col].astype(float) # Si ya es numérica, solo cambiamos
df[norm_col] = scaler.fit_transform(df[norm_col])
```

```
In [ ]: # Codifica columna categoricas con funcion OrdinalEncoder
encoder_planta_proceso = OrdinalEncoder()
encoder_formato = OrdinalEncoder()

# Codificar la columna "Planta Proceso"
encoded_planta_proceso = encoder_planta_proceso.fit_transform(df[["Planta Pr"])

# Codificar la columna "Formato"
encoded_formato = encoder_formato.fit_transform(df[["Formato"]])

# Asignar los valores codificados a las columnas en df_total
df[["Planta Proceso"]] = encoded_planta_proceso
df[["Formato"]] = encoded_formato
```

```
In [ ]: # Supongamos que 'scaler' es tu escalador (por ejemplo, una instancia de StandardScaler)
scaler = ... # tu escalador ajustado

# Guardar el escalador a un archivo
with open('scaler.pkl', 'wb') as archivo:
    pickle.dump(scaler, archivo)

# Cargar el escalador desde el archivo
with open('scaler.pkl', 'rb') as archivo:
    scaler_cargado = pickle.load(archivo)

# Ahora puedes usar 'scaler cargado' para escalar nuevos datos
```

```
In [ ]: # Supongamos que 'scaler' es tu escalador (por ejemplo, una instancia de StandardScaler)
scaler = ... # tu escalador ajustado

# Guardar el escalador a un archivo
with open('scaler.pkl', 'wb') as archivo:
    pickle.dump(scaler, archivo)

# Cargar el escalador desde el archivo
with open('scaler.pkl', 'rb') as archivo:
    scaler_cargado = pickle.load(archivo)

# Ahora puedes usar 'scaler cargado' para escalar nuevos datos
```

```
In [ ]: # Seleccionar las variables de entrada y la variable objetivo #'Planta Procesadora de Maiz'
X = df[['Peso Total (Kg)', 'Triticale', 'Maíz', 'Afrecho de trigo', 'Afrecho de cebada', 'Afrecho de maíz']]
```

```

'Carbonato de calcio', 'Subproducto avena', 'Arvejas ingentec', 'Cáscara
"Planta Proceso_Línea 2","Formato_Harinado", "Formato_Pellet 3mm", "Form

y = df['energia molino']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

```

In []:

```

# Inicializar el modelo de regresión de perceptrón multicapa
mlp = MLPRegressor(activation="relu", solver="adam", hidden_layer_sizes=(128, 64, 32))
# Entrenar el modelo
mlp.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = mlp.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
max_err = max_error(y_test, y_pred)

# Mostrar el error absoluto medio en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

In []:

```

## Mejores parámetros:
{'validation_fraction': 0.1, 'tol': 0.0001, 'solver': 'sag',
 'max_iter': 1000, 'learning_rate_init': 0.001, 'learning_rate': 'constant',
 'beta_2': 0.999, 'beta_1': 0.9, 'alpha': 0.001, 'activation': 'relu'}
Mejor puntuación: 0.6935463926242363
# 8 semanas: Error on the test set: (0.8014721906889025, 233.83162807206585,

```

In []:

```

# Guardar el modelo en un archivo
with open('mlp_model.pkl', 'wb') as file:
    pickle.dump(mlp, file)

# Cargar el modelo desde el archivo
with open('mlp_model.pkl', 'rb') as file:
    loaded_mlp = pickle.load(file)

```

In []:

```

# Crear el modelo SVR
svr_model = SVR(kernel='rbf', C=10.0, epsilon=0.1)

# Entrenar el modelo
svr_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_svr = svr_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_svr)
mse = mean_squared_error(y_test, y_pred_svr)
r2 = r2_score(y_test, y_pred_svr)

```

```

max_err = max_error(y_test, y_pred_svr)

# Mostrar el error absoluto medio en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Entrenar el modelo
rf_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_rf = rf_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_rf)
mse = mean_squared_error(y_test, y_pred_rf)
r2 = r2_score(y_test, y_pred_rf)
max_err = max_error(y_test, y_pred_rf)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo GradientBoostingRegressor
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.5, ra

# Entrenar el modelo
gb_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_gb = gb_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_gb)
mse = mean_squared_error(y_test, y_pred_gb)
r2 = r2_score(y_test, y_pred_gb)
max_err = max_error(y_test, y_pred_gb)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo AdaBoost
ada_model = AdaBoostRegressor(n_estimators=100, learning_rate=0.5, random_st

# Entrenar el modelo
ada_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_ada = ada_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_ada)
mse = mean_squared_error(y_test, y_pred_ada)
r2 = r2_score(y_test, y_pred_ada)

```

```

max_err = max_error(y_test, y_pred_ada)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo Stochastic Gradient Boosting
sgb_model = GradientBoostingRegressor(
    n_estimators=100,
    learning_rate=0.5,
    subsample=0.8, # Submuestra del 80% para cada árbol
    random_state=42
)

# Entrenar el modelo
sgb_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_sgb = sgb_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_sgb)
mse = mean_squared_error(y_test, y_pred_sgb)
r2 = r2_score(y_test, y_pred_sgb)
max_err = max_error(y_test, y_pred_sgb)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo HistGradientBoostingRegressor
hgb_model = HistGradientBoostingRegressor(learning_rate=0.5, max_iter=100, r

# Entrenar el modelo
hgb_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_hgb = hgb_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_hgb)
mse = mean_squared_error(y_test, y_pred_hgb)
r2 = r2_score(y_test, y_pred_hgb)
max_err = max_error(y_test, y_pred_hgb)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: from xgboost import XGBRegressor

# Crear el modelo XGBoost
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.5, random_state=4

# Entrenar el modelo
xgb_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba

```

```

y_pred_xgb = xgb_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_xgb)
mse = mean_squared_error(y_test, y_pred_xgb)
r2 = r2_score(y_test, y_pred_xgb)
max_err = max_error(y_test, y_pred_xgb)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo CatBoost
catboost_model = CatBoostRegressor(n_estimators=100, learning_rate=0.3, random_state=42)

# Entrenar el modelo
catboost_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_catboost = catboost_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_catboost)
mse = mean_squared_error(y_test, y_pred_catboost)
r2 = r2_score(y_test, y_pred_catboost)
max_err = max_error(y_test, y_pred_catboost)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Crear el modelo LightGBM
lgbm_model = LGBMRegressor(n_estimators=100, learning_rate=0.5, random_state=42)

# Entrenar el modelo
lgbm_model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred_lgbm = lgbm_model.predict(X_test)

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred_lgbm)
mse = mean_squared_error(y_test, y_pred_lgbm)
r2 = r2_score(y_test, y_pred_lgbm)
max_err = max_error(y_test, y_pred_lgbm)

# Mostrar las métricas de evaluación en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: # Inicializar el modelo de regresión de perceptrón multicapa
mlp = MLPRegressor(activation="relu", solver="adam", hidden_layer_sizes=(128, 64, 32))

# Entrenar el modelo
mlp.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = mlp.predict(X_test)

```

```

# Evaluar el modelo
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
max_err = max_error(y_test, y_pred)

# Mostrar el error absoluto medio en el conjunto de prueba
print(f"Error on the test set: {r2, mse, mae, max_err}")

```

```

In [ ]: param_grid = param_grid = {
    'hidden_layer_sizes': [(128, 32, 16)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'max_iter': [500, 1000],
    'learning_rate_init': [0.001, 0.01],
    'learning_rate': ['constant', 'invscaling', 'adaptive'],
    'alpha': [0.001],
    'momentum': [0.9],
    'beta_1': [0.9],
    'beta_2': [0.999],
    'tol': [0.0001],
    'validation_fraction': [0.1],
    'random_state': [42]
}

mlp = MLPRegressor()

# Inicializar RandomizedSearchCV
# Puedes ajustar n_iter para controlar el número de combinaciones de parámetros
random_search = RandomizedSearchCV(estimator=mlp, param_distributions=param_grid,
                                     n_iter=100, cv=5, scoring='neg_mean_squared_error',
                                     random_state=42, verbose=1, n_jobs=-1)

# Entrenar el modelo con RandomizedSearchCV
random_search.fit(X_train, y_train)

# Mejores parámetros y puntuación
print("Mejores parámetros:", random_search.best_params_)
print("Mejor puntuación:", random_search.best_score_)

# Hacer predicciones con el mejor modelo
y_pred = random_search.predict(X_test)

# Evaluar el mejor modelo
# ... (aquí puedes calcular las métricas de evaluación como antes)

```

```

In [ ]: # Definir los parámetros para el grid search
param_grid = {
    'hidden_layer_sizes': [(128, 32, 16)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'max_iter': [500, 1000],
    'learning_rate_init': [0.001, 0.01],
    'learning_rate': ['constant', 'invscaling', 'adaptive'],
    'alpha': [0.001],
    'l1_ratio': [0.1, 0.5, 0.9],
    'penalty': ['l1', 'l2'],
    'warm_start': [True, False]
}

```

```

'momentum': [0.9],
'beta_1': [0.9],
'beta_2': [0.999],
'tol': [0.0001],
'validation_fraction': [0.1],
'random_state': [42]
}

mlp = MLPRegressor()

# Inicializar GridSearchCV
grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, n_jobs=-1,
# Entrenar el modelo con GridSearchCV
grid_search.fit(X_train, y_train)

# Mejores parámetros y puntuación
print("Mejores parámetros:", grid_search.best_params_)
print("Mejor puntuación:", grid_search.best_score_)

# Hacer predicciones con el mejor modelo
y_pred = grid_search.predict(X_test)

# Evaluar el mejor modelo
# ... (aquí podrías calcular las métricas de evaluación como antes)

```

In []: Mejores parámetros: {'validation_fraction': 0.1, 'tol': 0.0001, 'solver': 'sgd', 'max_iter': 500, 'learning_rate_init': 0.001, 'learning_rate': 'invscaling', 'hidden_layer_sizes': (128, 150, 32, 16), 'beta_2': 0.999}

In []:

```
In [ ]: import pandas as pd
```

```
In [ ]: # Cargando el archivo CSV
df_pp = pd.read_csv("datos entrenamiento PP.csv")
df_pp
```

```
In [ ]: # Filtrando las filas con valores de 'Peso Total (Kg)' entre 14500 y 15500
df_filtrado = df_pp[(df_pp['Peso Total (Kg)'] >= 14500) & (df_pp['Peso Total (Kg)'] <= 15500)]
df_filtrado = df_filtrado.sort_values(by='energia peletizadora', ascending=False)
df_filtrado = df_filtrado.reset_index(drop=True)
df_filtrado
```

```
In [ ]: # Filtrando las filas con formato "Pellet 3mm"
df_pp_3mm = df_pp[df_pp['Formato'] == 'Pellet 3mm']
df_pp_3mm = df_pp_3mm.reset_index(drop=True)
df_pp_3mm
```

```
In [ ]: # Filtrando las filas que contienen "Línea 1" en la columna "Planta Proceso"
df_pp_3mm_linea1 = df_pp_3mm[df_pp_3mm['Planta Proceso'] == 'Línea 1']

# Calculando el promedio ponderado de 'energia peletizadora' respecto a 'Peso Total'
total ponderado linea1 = (df_pp_3mm_linea1['energia peletizadora']) / df_pp_3mm_linea1.sum()

print("Promedio ponderado de energia peletizadora para Línea 1: ", total_ponderado)
```

```
In [ ]: # Filtrando las filas que contienen "Línea 2" en la columna "Planta Proceso"
df_pp_3mm_linea2 = df_pp_3mm[df_pp_3mm['Planta Proceso'] == 'Línea 2']

# Calculando el promedio ponderado de 'energia peletizadora' respecto a 'Peso Total'
total ponderado linea2 = (df_pp_3mm_linea2['energia peletizadora']) / df_pp_3mm_linea2.sum()

print("Promedio ponderado de energia peletizadora para Línea 2: ", total_ponderado)
```

```
In [ ]: # Filtrando las filas con formato "Pellet 6mm"
df_pp_6mm = df_pp[df_pp['Formato'] == 'Pellet 6mm']
df_pp_6mm = df_pp_6mm.reset_index(drop=True)
pd.set_option('display.max_rows', None)
df_pp_6mm
```

```
In [ ]: # Filtrando las filas que contienen "Línea 1" en la columna "Planta Proceso"
df_pp_6mm_linea1 = df_pp_6mm[df_pp_6mm['Planta Proceso'] == 'Línea 1']

# Calculando el promedio ponderado de 'energia peletizadora' respecto a 'Peso Total'
total ponderado linea1 = (df_pp_6mm_linea1['energia peletizadora']) / df_pp_6mm_linea1.sum()

print("Promedio ponderado de energia peletizadora para Línea 1: ", total_ponderado)
```

```
In [ ]: # Filtrando las filas que contienen "Línea 2" en la columna "Planta Proceso"
df_pp_6mm_linea2 = df_pp_6mm[df_pp_6mm['Planta Proceso'] == 'Línea 2']

# Calculando el promedio ponderado de 'energia peletizadora' respecto a 'Peso Total'
total ponderado linea2 = (df_pp_6mm_linea2['energia peletizadora']) / df_pp_6mm_linea2.sum()
```

```

print("Promedio ponderado de energia peletizada para Línea 2: ", total_pon
In [ ]: # Filtrando las filas con formato "Pellet 8mm"
df_pp_8mm = df_pp[df_pp['Formato'] == 'Pellet 8mm']
df_pp_8mm = df_pp_8mm.reset_index(drop=True)
pd.set_option('display.max_rows', None)
df_pp_8mm

In [ ]: # Filtrando las filas que contienen "Línea 1" en la columna "Planta Proceso"
df_pp_8mm_linea1 = df_pp_8mm[df_pp_8mm['Planta Proceso'] == 'Línea 1']

# Calculando el promedio ponderado de 'energia peletizada' respecto a 'Peso'
total_ponderado_linea1 = (df_pp_8mm_linea1['energia peletizada'] / df_pp_8mm['peso']).sum()

print("Promedio ponderado de energia peletizada para Línea 1: ", total_pon
In [ ]: # Filtrando las filas que contienen "Línea 2" en la columna "Planta Proceso"
df_pp_8mm_linea2 = df_pp_8mm[df_pp_8mm['Planta Proceso'] == 'Línea 2']

# Calculando el promedio ponderado de 'energia peletizada' respecto a 'Peso'
total_ponderado_linea2 = (df_pp_8mm_linea2['energia peletizada'] / df_pp_8mm['peso']).sum()

print("Promedio ponderado de energia peletizada para Línea 2: ", total_pon

```