UNIVERSIDAD DE CONCEPCIÓN
Facultad de Ingeniería
Departamento de Ingeniería Metalúrgica

PROFESOR PATROCINANTE
Dr. Leopoldo Gutiérrez B.
INGENIERO SUPERVISOR
Jorge Cortínez C.

PREDICCIÓN DEL COMPORTAMIENTO DEL ESPESADOR DE RELAVES HRT 4001 EN FUNCIÓN DE LOS PARÁMETROS DE LA PULPA DE ALIMENTACIÓN Y REOLOGÍA EN COMPAÑÍA MINERA DOÑA INÉS DE COLLAHUASI

NATHALIA ANDREA OLIVERA BASCUR

Informe de Memoria de Título
para optar al Título de

Ingeniero Civil Metalúrgico

Octubre 2024

Dedicada a Rosa y Emilia,

Todo lo que soy y lo que seré es para ustedes.

Resumen

El presente proyecto de memoria de título expone el diseño de un modelo predictivo para el comportamiento del espesador de relaves HRT 4001 ubicado en la Planta Concentradora Ujina de Compañía Minera Doña Inés de Collahuasi, utilizando modelos de Machine Learning implementados con lenguaje de programación *Python*. Adicionalmente, con el modelo predictivo, se pueden identificar aquellas variables que permitan optimizar el funcionamiento del espesador y recuperación de agua.

Los tres modelos examinados corresponden a: Regresión Polinomial Multivariable, Redes Neuronales Artificiales (RNA) y Random Forest (RF). Los últimos dos modelos fueron modelados con la herramienta *GridSearchCV()* para crear modelos con el mejor desempeño posible. Las variables utilizadas para predecir el porcentaje de sólidos en la pulpa de descarga y reología (viscosidad y yield stress) incluyeron variables mineralógicas de la pulpa de alimentación (pH, nivel de arcillas y granulometría) y variables operacionales (flujo de alimentación, flujo de descarga, dosis de floculante, nivel de interfase, nivel de cama y torque).

Las métricas de rendimiento empleadas para comparar los modelos corresponden al Error Absoluto Medio (MAE), Raíz del Error Cuadrático Medio (RMSE) y el Coeficiente de determinación (R^2), que son comúnmente utilizadas en Machine Learning. Los resultados muestran que el modelo Random Forest produce mejores valores paras las tres variables objetivo, con un MAE de 0,042 para el porcentaje de sólidos en la descarga, 1,641 para la viscosidad y 7,363 para el yield stress, y un R^2 del 0,986, 0,906 y 0,981 respectivamente.

De los resultados se concluye que las características más influyentes corresponden a las arcillas y el nivel de cama. Aunque las características de la pulpa de alimentación no pueden ser modificadas por el proceso de espesamiento, el modelo predictivo permite experimentar con las demás variables, ayudando a identificar los ajustes necesarios para alcanzar los resultados objetivos, mientras se informa la reología.

Finalmente, como el modelo predictivo depende de los datos en línea, se sugiere una revisión de los sensores, principalmente del sistema de rastreo del tamaño de partículas, y la ampliación del conjunto de datos a lo largo del tiempo, entrenando el modelo con data futura, permitiendo que no pierda así su poder predictivo.

Abstract

This project presents the design of a predictive model for the behavior of the HRT 4001 tailings thickener located at the Ujina Concentrating Plant of Compañía Minera Doña Inés de Collahuasi, using Machine Learning models implemented with Python programming language. Additionally, with the predictive model, it is possible to identify variables that can optimize the thickener's performance and water recovery.

The three models examined correspond to: Multivariable Polynomial Regression, Artificial Neural Networks (ANN), and Random Forest (RF). The last two models were tuned using the GridSearchCV() tool to create models with the best possible performance. The variables used to predict the solids percentage in the discharge pulp and rheology (viscosity and yield stress) included mineralogical variables of the feed pulp (pH, clay level, and particle size) and operational variables (feed flow, discharge flow, flocculant dosage, interface level and torque).

The performance metrics used to compare the models correspond to the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2), which are commonly used in Machine Learning. The results show that the Random Forest model produces better values for the three target variables, with an MAE of 0.042 for the solids percentage in the discharge, 1.641 for viscosity, and 7.363 for yield stress, and an R^2 of 0.986, 0.906, and 0.981, respectively.

From the results, it is concluded that the most influential features are clay and bed level. Although the characteristics of the feed pulp cannot be modified by the thickening process, the predictive model allows experimentation with other variables, helping to identify the necessary adjustments to achieve target results while reporting rheology.

Finally, since the predictive model depends on online data, it is suggested to review the sensors, mainly the particle size tracking system, and to expand the dataset over time, training the model with future data to ensure it retains its predictive power.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

La escasez de recursos hídricos se ha convertido en uno de los desafíos más importantes en Chile y el cambio climático es un factor que agrava la escasez de agua en muchas regiones, debido a la falta de precipitaciones y el aumento de temperaturas, intensificando la menor disponibilidad de agua, especialmente en la zona norte del país, donde se concentra mayormente la industria minera.

El desarrollo de una minería sustentable implica hacer uso eficiente de este recurso. El uso de agua es imprescindible en la minería del país, especialmente en la producción de cobre y molibdeno, que consume en sus procesos productivos aproximadamente 19 m^3/s de agua, lo que reafirma que la minería es una actividad que necesita del recurso hídrico en cada etapa del proceso minero, desde la extracción hasta el procesamiento y la disposición de residuos [1].

La última proyección de demanda de agua en la minería del cobre de la Comisión Chilena del Cobre del año 2023 indica, tal como se presenta en la Figura 1, que para el año 2034 se espera que el nivel de consumo de agua a nivel nacional sea de 23.7 m^3/s, con una tasa de crecimiento promedio anual de 2.3%, con una disminución de un 39% de la demanda de agua continental pero un aumento del 157% de la demanda de agua de mar, con respecto al 2022, teniendo en consideración que la disminución en las leyes de los minerales de cobre hace necesaria una mayor cantidad de agua en su procesamiento para mantener el nivel de producción del país.

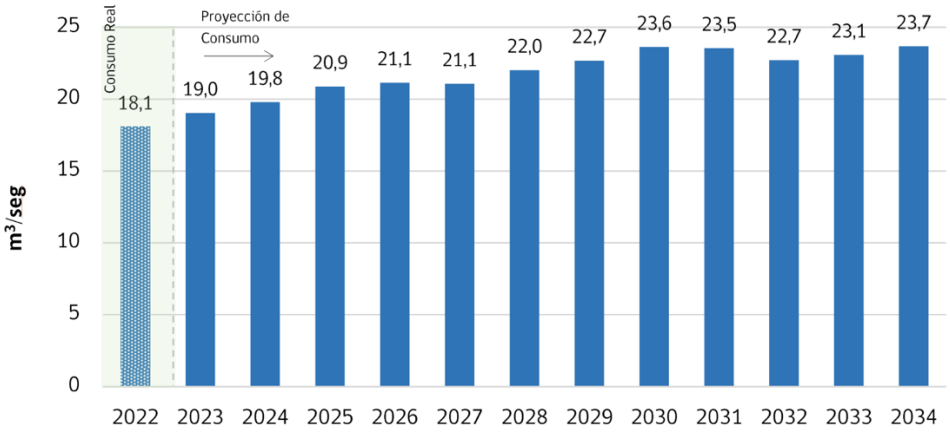


Figura 1. Proyección demanda de agua en la minería del cobre 2023 – 2034. COCHILCO, 2023. Extraído de [1].

Las plantas concentradoras, que corresponden al área de procesamiento de minerales, que involucra la conminución del mineral, flotación, clasificación y posterior espesamiento, comprende al mayor consumo de agua respecto al volumen total de consumo en la industria minera del país. Se proyecta que, para el 2034, la planta concentradora representará el 82.9% del consumo de agua total, mientras la hidrometalurgia solo un 4.3% [1].

Para su optimización, la estrategia más utilizada es la recirculación y reúso. Se han perfeccionado los procesos de espesamiento y filtración, además de la exploración de tecnologías más eficientes [2]. Por lo tanto, el espesamiento de relaves es fundamental para maximizar la reutilización del recurso hídrico, generando una pulpa con mayor concentración de sólido, que es enviada al tranque de relaves y agua clara recirculada al proceso productivo, disminuyendo los valores de make – up de agua fresca.

En este contexto es de suma importancia predecir el comportamiento de estos equipos, principalmente el porcentaje de sólidos de la pulpa de descarga, permitiendo un mejor control operacional y toma de decisiones oportunas.

2. Objetivos

2.1 Objetivo General

Generar un modelo predictivo en base a variables de la pulpa de alimentación, condiciones operacionales y reología para predecir el comportamiento del espesador de relaves HRT 4001.

2.2 Objetivos Específicos

Identificar las variables de entrada y salida que influyen en el comportamiento de los espesadores de relaves, específicamente el espesador HRT 4001 de CMDIC.

Identificar situaciones operacionales que afectan la operación del espesamiento de relaves analizando el comportamiento histórico del proceso a través de datos obtenidos desde la plataforma de *PI System* y el reómetro en línea *'KRheo'* instalado por la empresa Konatec.

Pre – procesar y filtrar los datos obtenidos para la generación de modelos predictivos mediante el uso de algoritmos de Machine Learning.

- Evaluar y comparar los distintos modelos identificando el que posea mejor desempeño.

3. Antecedentes

3.1 Antecedentes de la empresa

Compañía Minera Doña Inés de Collahuasi (CMDIC) se dedica a la extracción y producción de concentrados de cobre y molibdeno, logrando excelencia y calidad en la operación, en la gestión de los recursos humanos, control de pérdidas y cumpliendo con las normas de protección ambiental, siendo una de las cuatro principales productoras de molibdeno en Chile y una de las seis productoras de cobre más grande del mundo, con uno depósito de 10.380 millones de toneladas de cobre.

Ubicada en la Región de Tarapacá, sus instalaciones en el Área Cordillera incluyen los yacimientos Rosario y Ujina a una altitud promedio de 4.400 metros sobre el nivel del mar, una zona andina que se caracteriza por tener clima lluvioso en verano y nevadas ocasionales en invierno. Desde su planta concentradora Ujina, un sistema de mineroductos, de 7 y 8 pulgadas de 203 km de extensión transporta el concentrado colectivo de cobre y molibdeno como pulpa hasta el Terminal Marítimo Collahuasi en Puerto Patache, a 65 km al sur de la ciudad de Iquique, donde se encuentra la planta de molibdeno y filtrado de concentrado.

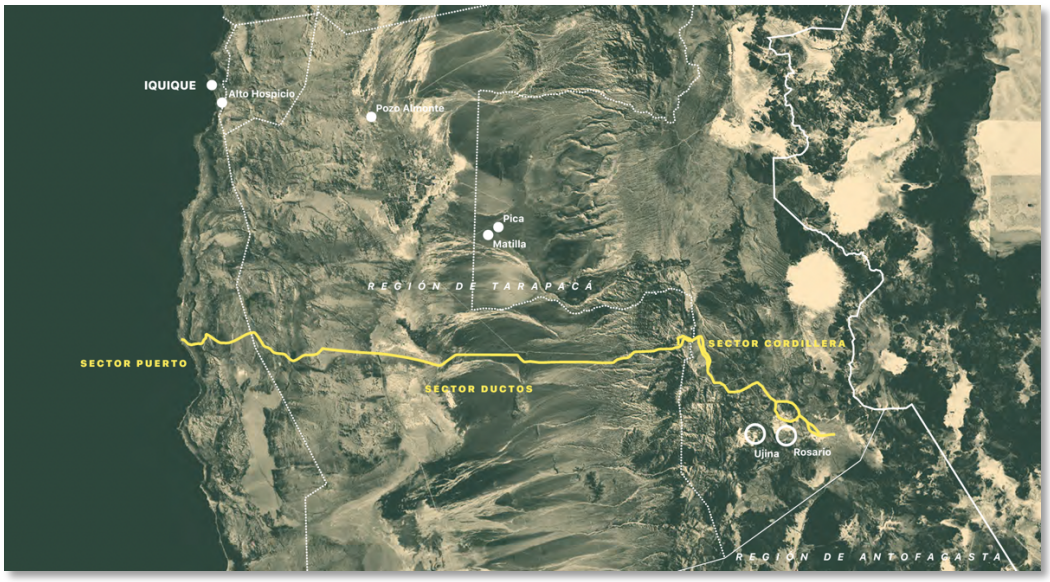


Figura 2. Ubicación Compañía Minera Doña Inés de Collahuasi.

CMDIC es una sociedad contractual minera, cuyos accionistas son Anglo American plc (44%), Glencore (44%) y Japan Collahuasi Resources B.V. (12%), los que están representados en su directorio.

Dentro de los objetivos estratégicos de la compañía se encuentra la transformación digital, donde se identifica cualquier oportunidad de digitalización que permita aumentar los niveles de eficiencia. Esta estrategia apunta a aprovechar los datos generados por cada proceso y aplicarlos mediante algoritmos que permitan predecir resultados, reducir riesgos, y agregar valor en términos de optimización de los costos, productividad, seguridad y sustentabilidad [3].

3.2 Descripción de la operación CMDIC

Actualmente, la línea de sulfuros se encuentra operativa, mientras que la línea de óxidos se encuentra detenida y en proceso de mantención, a la espera de mejores condiciones de mercado para su activación.

Por lo tanto, el mineral extraído desde la mina Rosario, que corresponde a un yacimiento abierto, se reduce inicialmente en el chancador primario, que se ubica en el área del rajo. Este chancador es del tipo giratorio y el mineral chancado tiene un tamaño máximo de 300 mm y es enviado mediante una correa transportadora hacia la zona de acopio del mineral, denominado *Stock Pile*, que se ubica en el área de la Planta Concentradora Ujina. Para continuar con la reducción de tamaño del mineral y la separación del cobre sulfurado de la ganga, el mineral es tratado por 3 líneas de molienda, las que alimentan la planta concentradora de sulfuros. Este proceso productivo se presenta en la Figura 3.

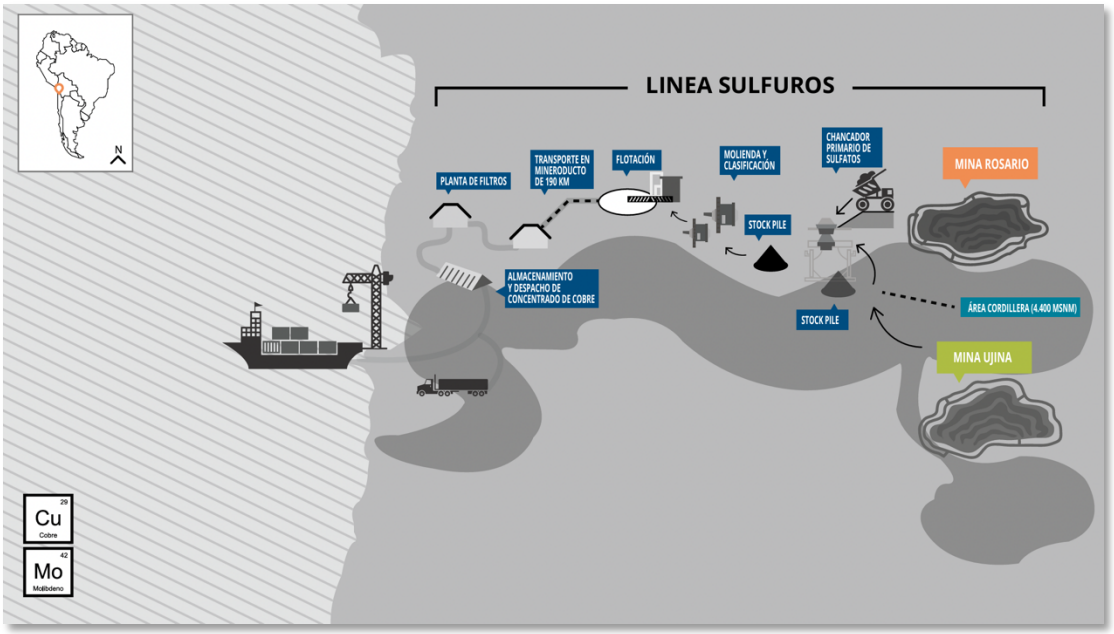


Figura 3. Proceso productivo de línea de sulfuros de CMDIC.

De las 3 líneas de molienda, 2 líneas están compuestas cada una por un molino SAG y un molino de bolas, en cada línea. La tercera línea de molienda compone un molino SAG y dos molinos de bolas. Esta operación es asistida por 8 baterías de hidrociclones.

Las tres líneas de molienda alimentan el circuito de flotación primaria o *rougher*, que corresponden desde la línea 1 a la línea 5, y desde la línea D a la G, cada línea con 9 celdas convencionales cada una. Las colas de esta etapa se descartan inmediatamente como relave, mientras que el concentrado alimenta el circuito de clasificación y remolienda, que corresponden a 4 baterías de hidrociclones y 8 molinos verticales, que alimentan la primera etapa de flotación de limpieza o *cleaner* y *scavenger*, que abarcan las líneas 6 y 7, y desde la línea A hasta la C, cada línea con 9 celdas convencionales. Las colas de la etapa *scavenger* son enviadas directamente al espesamiento de relaves y el concentrado es recirculado a la etapa de remolienda y clasificación, mientras que el concentrado de la primera etapa de limpieza es transferido a la segunda limpieza, compuesto por 10 columnas, 6 rectangulares y 4 redondas. Las colas de estas celdas son transferidas a la etapa *scavenger* de la segunda limpieza, que corresponden a 4 celdas *Jameson*, mientras que el concentrado corresponde a una parte del concentrado final. Finalmente, las colas de las celdas *Jameson* son recirculadas a la primera etapa de limpieza y el concentrado corresponde a la otra parte del concentrado final de la flotación colectiva de cobre y molibdeno, que es transportado a Puerto Patache donde se realiza la flotación selectiva. Cabe destacar que, el 97% de la alimentación de la planta concentradora corresponde a relaves, que son transportados a los

espesadores de relaves con el objetivo de recuperar agua y ser reutilizada en los procesos de concentración.

En la actualidad CMDIC cuenta con 9 espesadores de relaves con distintas características: 3 espesadores HRT de 125 m de diámetro (HRT 2001, HRT 3001 y HRT 4001), 2 DOE de 60 m de diámetro (DOE 1011 y DOE 1012) y 4 espesadores convencionales de 43 m de diámetro, los últimos fueron sacados de operación a mediados del 2017. En la Figura 4 se muestra la vista aérea de los espesadores mencionados.

Figura 4. Vista aérea de los espesadores de relave en Área Cordillera de CMDIC.

Todo el proceso productivo explicado se presenta en el diagrama de la Planta Concentradora Ujina en la Figura 5.

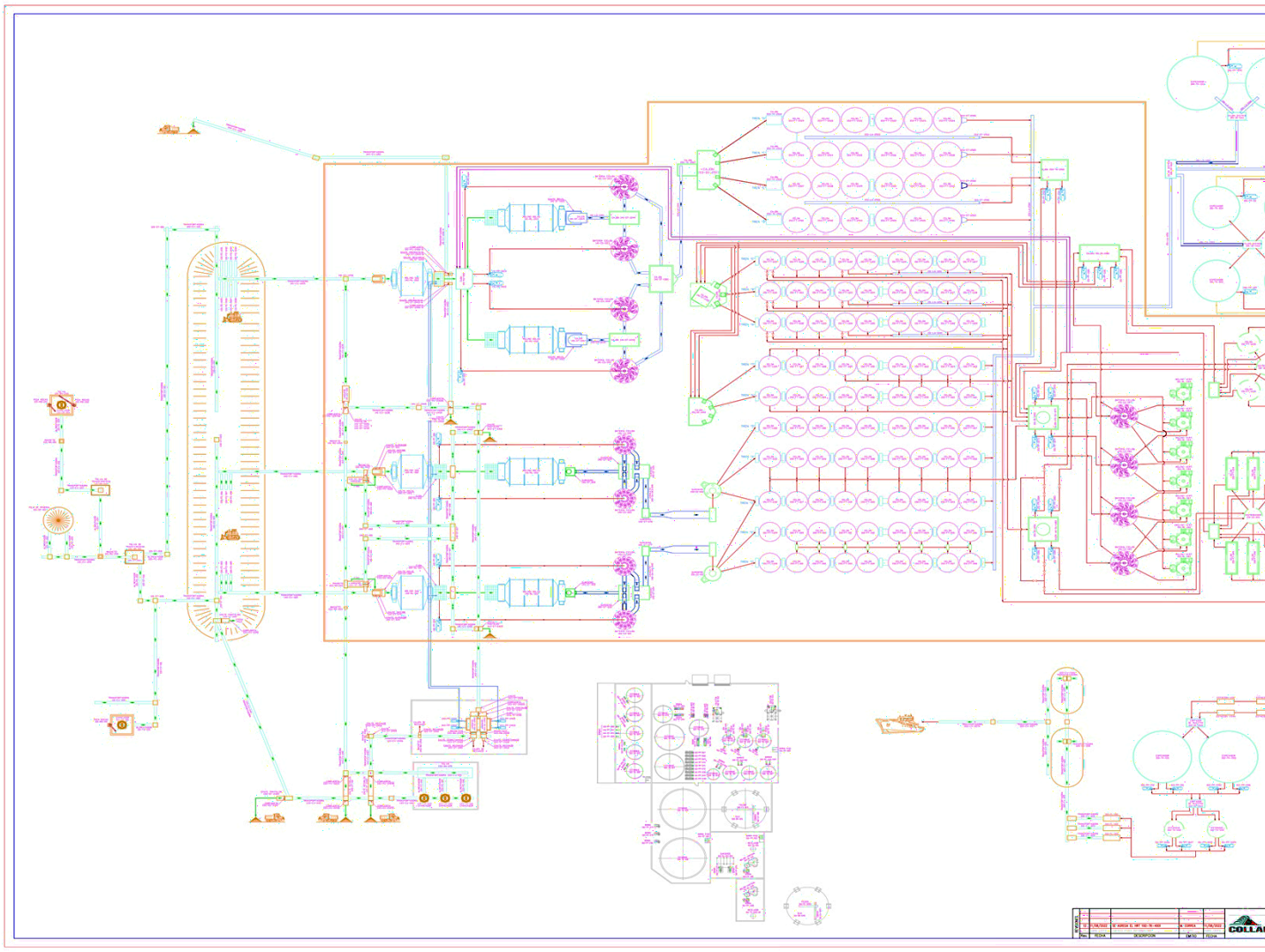


Figura 5. Diagrama Planta Concentradora de Compañía Minera Doña Inés de Collahuasi.

Finalmente, la pulpa espesada en los espesadores de relaves se deposita en el tranque de relaves. El muro del tranque de relaves se construye con la metodología “aguas abajo”, lo que permite asegurar la estabilidad del muro frente a requerimientos sísmicos y los materiales utilizados en su cuerpo principal son provenientes de los botaderos mina. Su muro principal tiene una extensión de 5.900 metros de largo y una altura máxima de 78 metros.

El agua es un recurso estratégico para el desarrollo de los procesos de CMDIC, y su uso eficiente es uno de los pilares de su estrategia de sustentabilidad, buscando la máxima eficiencia en su utilización. El agua utilizada para el procesamiento del mineral, aproximadamente el 79% es agua recuperada de los espesadores y de la laguna de clarificación de relaves, además, el make – up de agua fresca al cierre 2023 alcanzó a 0.45 m^3/ton.

3.3 Detalles del equipo HRT 4001 de CMDIC

El equipo estudiado para el presente proyecto de memoria corresponde al espesador de relaves HRT 4001 de tracción periférica, considerado el espesador más grande del mundo. Su construcción comenzó a fines del año 2019 en el sector Ujina, concluyendo en el año 2021.

Los parámetros de diseño y dimensiones del equipo se encuentran en la Tabla 1.

Tabla 1. Parámetros metalúrgicos de diseño del espesador de relaves HRT 4001 de CMDIC.

Parámetro	Diseño del equipo	Nominal
Tratamiento de sólidos, t/h	3.750	2.900
Flujo pulpa de alimentación, m^3/h	11.208	10.850
% Sólidos en alimentación, %p/p	27 – 32	
Densidad de pulpa de alimentación, ton/m^3	1,2	
Gravedad específica del sólido, ton/m^3	2,64	
pH pulpa de alimentación	10 – 10,5	10
% Sólidos en la descarga, %p/p	62	59
Flujo de pulpa de descarga, m^3/h	3.719	3.460
Flujo de agua recuperada, m^3/h	7.490	-
% Sólidos dilución pulpa de alimentación	5,5 – 10%	5,5
Torque instalado, Ft – lbs	10.000.000	
Torque operacional, %	<40	
Diámetro, m	125	
Altura pared lateral, m	3,0	
Pendiente piso, °	14	
Profundidad en el centro, m	19,58	
Volumen estanque, m^3	100.858	
Diámetro feedwell, m	20,0	

En la operación, la pulpa de alimentación se diluye a una concentración de sólidos cercana al 5,5% utilizando dos agitadores del sistema P – DUC, que garantiza un nivel de dilución óptimo y constante. Este sistema, utiliza agua clara del mismo estanque, donde un agitador de baja energía y alta eficiencia diluye la pulpa hasta el porcentaje de sólidos óptimo, y la alimentación diluida se mezcla con el floculante, que corresponde al SNF 604, antes de llegar a la bandeja de alimentación.

En cuanto a la reología de la pulpa de descarga del espesador, se consideran valores óptimos aproximados de yield stress de 250 dinas/cm^2 y constante de viscosidad plástica de Bingham de 20 centipoise, para no tener problemas de transporte hacia el tranque de relave, sin embargo, se excede a valor de 330 dinas/cm^2 y 25 centipoise. Para medir estas variables reológicas de la pulpa, desde una de las dos líneas de descarga posee el espesador de relaves HRT 4001, el equipo cuenta con el reómetro suministrado por la empresa Konatec, de nombre comercial *'KRheo'*, acondicionado para trabajar a 4.500 metros sobre el nivel del mar. Este equipo se muestra en la Figura 6.

Figura 6. Reómetro en línea 'KRheo' de la empresa Konatec.

4. Estado del Arte

4.1 Relaves

El relave es un sólido finamente molido, transportado en forma de pulpa, que se descarta en operaciones mineras. Están compuestos por minerales que no tienen interés económico, sustancias químicas, materia orgánica y efluentes, generados después del proceso de reducción de tamaño y concentración por procesos de flotación, donde se extraen los minerales de interés [4].

Este material, muy pobre en cobre, pasa por procesos de separación sólido – líquido para recuperar el agua que es recirculada al proceso minero y la pulpa descargada debe ser depositada de forma segura y ambientalmente responsable.

4.2 Espesamiento

El espesamiento es un proceso de separación sólido – liquido cuyo objetivo es la recuperación de agua (fase líquida) de la pulpa (fase sólida), que posee mayor concentración de sólidos en comparación con la pulpa alimentación. Es una operación continua que utiliza la fuerza de gravedad para separar las partículas del agua y se lleva a cabo en grandes tanques cilíndricos llamados espesadores, mediante la sedimentación de partículas como flóculos [5].

El espesador es el principal equipo utilizado para la separación sólido – líquido en la industria minera. Este equipo consta de cuatro zonas principales: una zona superior de agua clara, una zona de sedimentación debajo del agua clara, una zona de compresión y una zona de sedimento en la parte inferior, con una concentración constante y final, donde se encuentra la acción de las rastras [6].

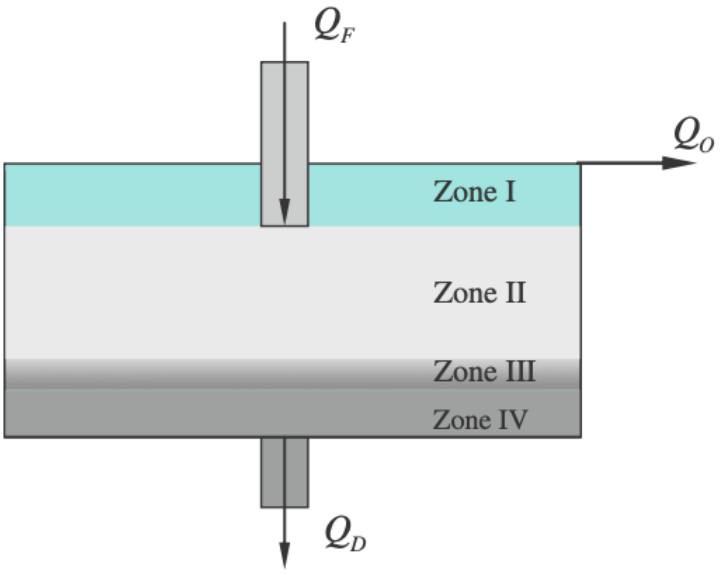


Figura 7. Esquema de zonas de un espesador convencional. Zona I: zona de agua clara, Zona II: zona de sedimentación, Zona III: zona de comprensión Y Zona IV: Zona de sedimento y acción de las rastras. Extraído de [5].

Con el tiempo se han hecho más grandes y de diferentes materiales, pero los elementos de su funcionamiento siguen siendo los mismos. Por lo tanto, su apariencia no ha cambiado mucho desde la invención del primer espesador Dorr en 1905.

Estos equipos, como se muestra en la Figura 7, se componen de un estanque cilíndrico que constituye el cuerpo del espesador, la bandeja de alimentación o *feedwell*, diseñado para mezclar la pulpa de alimentación con el floculante, además de diluir la pulpa de alimentación y distribuirla uniformemente en el espesador, la canaleta de recuperación de agua u *overflow* en la periferia del estanque, donde el agua recuperara se evacúa lentamente para evitar arrastrar partículas finas, y las rastras que cumplen la función de arrastrar el sedimento desde el fondo del estanque hacia la salida de descarga y la abertura de descarga o *underflow*.

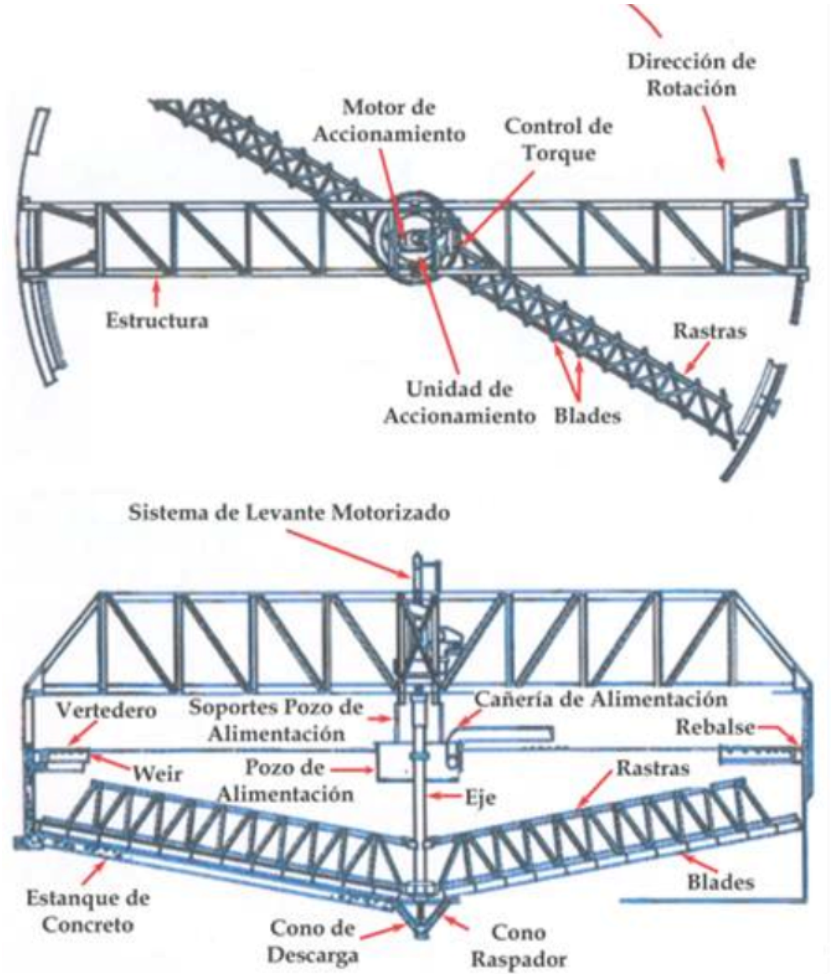


Figura 8. Espesador convencional. Extraído de [8].

Existen diferentes tipos de espesadores, los cuales mantienen la misma forma estructural y los mismos elementos mencionados, pero poseen detalles de diseño, como la ubicación del *feedwell,* dimensiones y pendiente, que diferencian su desempeño [6].

- **Espesador Convencional:** se caracteriza por su bandeja de alimentación ubicada en la parte superior del equipo, donde la pulpa de alimentación se mezcla con parte del agua recuperada, diluyéndola.

- **Espesador de Alta Capacidad:** tiene una bandeja de alimentación más profunda que el espesador convencional, que descarga la pulpa de alimentación bajo el nivel de sedimento, y se distingue por la ausencia de la zona de sedimentación.

- **Espesador de Alta Densidad:** es un espesador convencional o de alta capacidad, pero con mayor altura, lo que permite una concentración de descarga muy alta.

- **Espesador de Pasta:** comparado con los espesadores convencionales, tiene una mayor altura y una pendiente más pronunciada, logrando una alta concentración de sólidos en la descarga, formando una pasta densa [7].

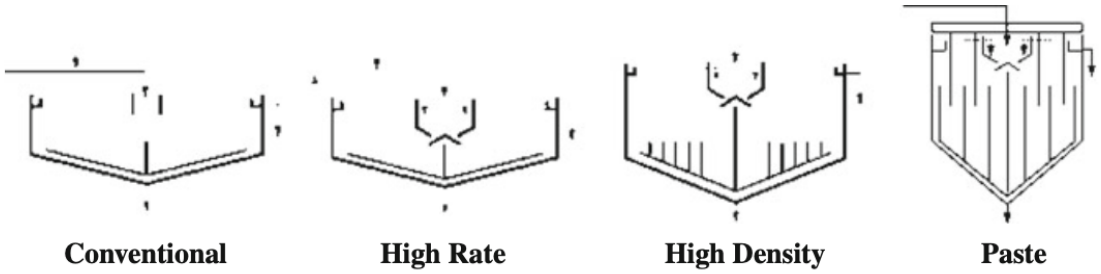


Figura 9. Representación de diferentes tipos de espesadores. Extraído de [5].

El proceso de espesamiento implica la superposición de dos fenómenos. El asentamiento de partículas o flóculos en un fluido gracias a la fuerza de gravedad, es decir, la sedimentación, lo que resulta en la formación de una cama. Durante la sedimentación, las partículas interactúan entre sí, obstaculizando su trayectoria y disminuyendo la velocidad de asentamiento que tendrían individualmente. Este proceso culmina cuando los flóculos llegan al fondo del espesador y comienzan a compactarse.

Al llegar al fondo, los flóculos descansan unos sobre otros, comprimiendo los flóculos inferiores y liberando el agua que se encuentra en su interior. Este fenómeno de eliminación de agua por compresión se conoce como consolidación [6].

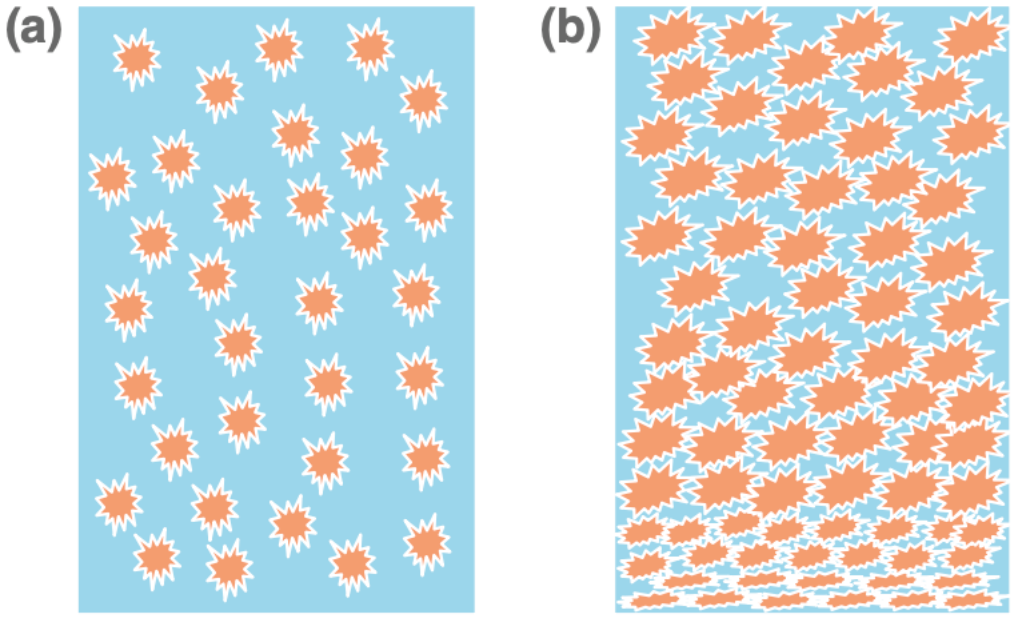


Figura 10. Modelo físico de a) sedimentación y b) consolidación. Extraído de [6].

Mediante ambos mecanismos, que pueden acelerarse con la adición de reactivos llamados floculante, se obtiene una pulpa con mayor concentración de sólidos.

Los floculantes son cadenas largas de polímeros, de alto peso molecular, que se fijan a las partículas sólidas formando flóculos de mayor peso que sedimentan a mayor velocidad, reduciendo el tiempo de separación sólido – líquido. Pueden ser polisacáridos naturales o materiales acrílicos sintéticos. Los floculantes sintéticos tienen la ventaja de que pueden diseñarse según los requisitos industriales, en términos de peso molecular más altos, y se dividen en polímeros no iónicos, aniónicos o catiónicos.

Existe una razón óptima entre la dosis de floculante y la cantidad de sólido, razón que, por encima de ella, el floculante ya no es adsorbido. Además, la floculación asistida por estos polímeros se divide en dos etapas: la adsorción de las cadenas sobre las partículas y luego el aglutinamiento y floculación.

El proceso de mezclado en el feedwell del espesador aumenta la interacción entre el floculante y las partículas sólidas, facilitando la adsorción. Este fenómeno entre el floculante y la superficie de una partícula sólida puede ocurrir a través de distintos mecanismos [3]:

- **Interacción electrostática:** atracción entre el polímero floculante y una superficie de carga opuesta.

- **Enlaces de hidrógeno:** minerales que contienen grupos hidroxilos en su estructura forman puentes de hidrógeno con polímeros que contienen grupos amida, carboxilo o hidroxilo.

- **Interacción hidrofóbica:** se genera entre segmentos hidrófobos de la cadena polimérica y partículas apolares.

- **Enlaces iónicos:** en algunas ocasiones, polímeros y superficies con igual carga pueden interactuar a través de una especie intermediaria. Si hay presencia de ciertos iones divalentes como el calcio, es posible generar una adhesión, debido a que estos iones divalentes se enlazan con los grupos carboxílicos del polímero y servir como unión.

Sin embargo, durante el fenómeno de floculación actúan varios de estos mecanismos de forma simultánea.

4.3 Variables que afectan en el espesamiento de relave

4.3.1 Características de la pulpa de alimentación

- **Granulometría:** refiere al tamaño de partícula, el cual genera doble impacto en el proceso de espesamiento, tanto en la velocidad de sedimentación como en las propiedades reológicas de la pulpa en la descarga del espesador. Cuando las partículas tienen un mayor tamaño, adquieren una mayor velocidad de sedimentación, esencial para una buena separación del fluido. Es por esto que, el espesamiento de relaves es difícil cuando se tratan partículas muy finas.

- **Minerales de arcillas:** se encuentran presentes en la mayoría de los yacimientos de cobre sulfurado. La presencia de estas especies minerales generan una disminución en la recuperación de agua, el aumento de viscosidad de la pulpa debido a su granulometría fina y la disminución de la velocidad de sedimentación de las partículas, lo que dificulta la operación del espesador y el transporte de pulpas.

- **pH:** principal efecto sobre el floculante, ya que cada floculante tiene un rango de trabajo óptimo. Por lo tanto, si cambia el pH puede ser necesario cambiar el floculante. Además, puede influir en la carga superficial de las partículas, afectando su estabilidad en suspensión, un pH adecuado puede evitar la aglomeración no deseada de las partículas, mejorando la eficiencia del espesamiento.

4.3.2 Variables operacionales

- **Dosis de floculante:** juega un papel crucial en el espesamiento de relaves, existiendo una razón óptima entre la cantidad de floculante y sólido que optimiza la eficiencia del proceso al permitir controlar la velocidad de sedimentación de las partículas al formar flóculos de mayor tamaño.

  La dosis de floculante depende el tipo de mineral y de la distribución de tamaño que ingresa al equipo, además su rendimiento depende del pH, pudiendo dejar de funcionar en caso de pH ácido. Si existe una sobredosificación, el floculante se adsorbe completamente en una partícula no permitiendo que se adsorba en otra, además puede llevar a la formación de flóculos demasiado grandes y densos, que pueden provocar generar problemas de torque o embancamiento, dificultando su operación.

- **Flujo de alimentación:** flujo volumétrico de pulpa que ingresa al espesador, controlar el caudal garantiza que el espesador funcione dentro de su capacidad de diseño, sin el riesgo de producir un embancamiento. Una alimentación controlada permite una sedimentación uniforme y eficiente.

- **Flujo de descarga:** permite controlar la altura del sedimento y el porcentaje de sólidos en la pulpa de descarga ya que, un menor flujo de descarga permite que los sólidos sedimenten más y se concentren en el fondo del espesador antes de ser descargados, favoreciendo la recuperación de agua y reduciendo la cantidad de líquido en la pulpa descargada. Este flujo debe ser ajustado de tal forma de equilibrarse con el flujo de alimentación.

- **Nivel de interfase:** cantidad de sedimento que se acumula en el fondo del espesador. Generalmente, un incremento en el nivel de la interfase, da como resultado un aumento del

porcentaje de sólidos en la descarga así como un aumento de presión en la base del espesador y valores de torque.

- **Nivel de cama:** se correlaciona con la presión de la cama, que orienta cuanto de altura de cama se está obteniendo, pero una cama alta no necesariamente significa que la presión de cama también será alta. Por ejemplo, si se suministra insuficiente floculante, se puede desarrollar una cama alta pero con baja densidad que puede ser detectado por los sensores de presión.

  Los niveles óptimos de interfase y cama se determinan mediante monitoreo constante, evaluación continua del rendimiento del espesador y concentración de sólidos en la descarga. Ajustes en el flujo de alimentación, flujo de descarga y dosificación del floculante son necesarios para mantener estos niveles en el rango adecuado.

- **Torque:** fuerza por unidad de distancia que deben tener las rastras del espesador para lograr que el sedimento espesado pueda llegar a la zona de descarga y corresponde a la demanda máxima que puede tener el motor del espesador sin dañar ninguno de sus elementos, por lo tanto, es la fuerza necesaria para mover las rastras dentro de los sólidos.

  Generalmente, los valores máximos de torque para los espesadores corresponden al 40%, cuando alcanza dicho valor se alerta en la operación. La primera acción a tomar cuando se alcanzan esos valores es incrementar el flujo de descarga hasta que el porcentaje de torque disminuya. El alto torque puede resultar del exceso de floculante y/o una sobrealimentación del espesador.

4.3.3 Reología

Corresponde al estudio de la deformación y el fluir de la materia bajo la influencia de fuerzas externas, busca entender como los materiales responden a tensiones aplicadas, ya sea mediante flujo, como un líquido, o deformación, como un sólido. Por lo tanto, el comportamiento reológico es crucial para la operación eficiente del espesamiento, ya que afecta la sedimentación y la capacidad de manejo de la pulpa de descarga, en especial si se busca una descarga con una alta concentración de sólidos [10].

Las principales características reológicas necesarias para describir las propiedades de la pulpa mineral son el yield stress y la viscosidad.

4.3.3.1 Yield stress

También llamado esfuerzo de fluencia, es el esfuerzo mínimo que debe aplicarse a una pulpa para que comience a fluir. Es decir, el umbral por encima del cual el material se comporta como un líquido y por debajo del cual se comporta como un sólido [11].

Las variables que influyen el yield stress son la concentración del sólido, tamaño y forma de las partículas, el pH y la naturaleza del mineral. A mayor porcentaje de sólidos, a la pulpa le cuesta más fluir, presentando un mayor yield stress.

4.3.3.2 Viscosidad

Refiere a la resistencia que presenta un fluido a fluir [11]. En el contexto del espesamiento de relave, la viscosidad afecta como fluye la pulpa a través del espesador y como los sólidos sedimentan.

De forma similar que el yield stress, a mayor porcentaje de sólidos, aumenta la viscosidad, ya que a concentraciones de sólidos más altas el movimiento de cualquier partícula conduce al movimiento de partículas cercanas y el material tiende a comportarse como un sólido.

Además, una viscosidad alta puede reducir la velocidad de sedimentación de las partículas, afectando en la eficiencia del espesador.

4.4 Machine Learning

El Machine Learning se centra en el estudio y construcción de sistemas que pueden aprender de datos sin ser programados explícitamente, utilizado para construir modelos, descubrir patrones y tendencias ocultas, descubriendo relaciones entre variables no sencillas de distinguir y predecir con ello resultados de interés [12].

Se clasifican en distintos tipos de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. En el aprendizaje supervisado el algoritmo aprende a partir de un conjunto de datos etiquetados, permitiendo al algoritmo predecir las variables objetivo para un conjunto de datos nuevo. En cambio, en el aprendizaje no supervisado, el algoritmo busca patrones en un conjunto de datos no etiquetados. En el aprendizaje por refuerzo, el algoritmo aprende a tomar decisiones mediante la interacción con un entorno dinámico [13].

Entre las técnicas de Machine Learning, la regresión y clasificación son métodos utilizados para abordar problemas predictivos. En el presente proyecto de memoria se utiliza aprendizaje supervisado con técnicas de regresión, ya que el modelo es alimentado con secuencias entrada – salida, y las salidas corresponden a valores continuos.

Todos los modelos de Machine Learning enfrentan el desafío del equilibrio entre sesgo y varianza. El sesgo se refiere a cuánto se desvían en promedio las predicciones de un modelo con respecto a los valores real, reflejando la habilidad del modelo para capturar la verdadera relación entre las variables predictoras y las variables respuesta. La varianza hace referencia a cuánto cambia el modelo en función de los datos utilizando en su entrenamiento. Idealmente, un modelo no debería cambiar demasiado ante pequeñas variaciones en los datos de entrenamiento, si esto ocurre, indica que el modelo está memorizando los datos en lugar de aprender las verdaderas relaciones entre las variables [12].

Conforme aumenta la complejidad del modelo, este dispone una mayor flexibilidad para adaptarse, lo que conlleva una reducción del sesgo y una mejora en su capacidad predictiva. Sin embargo, una vez alcanzado cierto nivel de complejidad surge el problema de sobreajuste o *overfitting*, que se presenta cuando el modelo se ajusta tanto a los datos de entrenamiento que es incapaz de predecir correctamente nuevas observaciones [12].

Por lo tanto, el modelo óptimo es aquel que logra el equilibrio adecuado entre sesgo y varianza [14].

Se examinan 3 modelos en particular, que corresponden a, Regresión Polinomial Multivariables, Redes Neuronales Artificiales y Random Forest [12]:

- **Regresión Polinomial Multivariable:** técnica de Machine Learning utilizada para modelar la relación entre múltiples variables dependientes y variables independientes. Corresponde a una extensión de la regresión lineal, pero capaz de capturar relaciones no lineales entre variables.

- **Redes Neuronales Artificiales (RNA):** modelo computacional inspirado biológicamente en las neuronas cerebrales capaz de reproducir el método de aprendizaje entrenado.

  Una red neuronal está compuesta por un conjunto de nodos interconectados que representan las neuronas, organizadas en capas. La capa de entrada corresponde al número de variables predictoras, las capas ocultan procesan la información a través de una serie de transformaciones, y la capa de salida representa las variables a predecir, conectadas entre sí por diferentes pesos que determinan la fuerza de conexión entre las neuronas.

  Para emular la reacción de excitación de una neurona se utilizan funciones de activación, cuyo objetivo es introducir no linealidad en el modelo, permitiendo a la red neuronal aprender y representar relaciones complejas entre los datos. Algunas funciones comunes incluyen la función sigmoide, la tangente hiperbólica (tanh) y la ReLU. Cuál utilizar depende del tipo de problema.

  El entrenamiento de las redes neuronales implica ajustar los pesos de la red para minimizar la diferencia entre las predicciones de la red y los valores reales. Además, existen distintas arquitecturas dentro de una red, como el perceptrón simple compuesta por un solo nivel de neuronas y el perceptrón multicapa, donde las neuronas se disponen en dos o más capas.

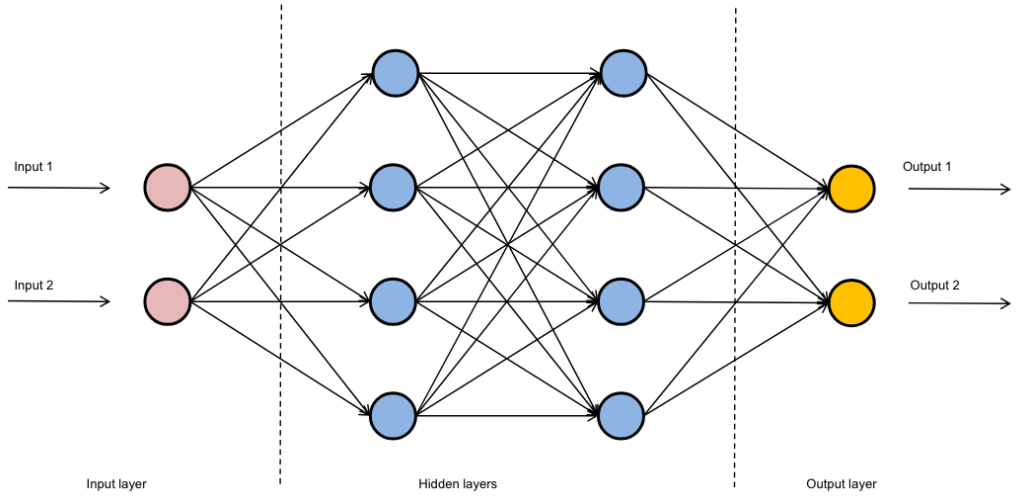


Figura 11. Representación de RNA perceptrón multicapa, con 2 variables de entrada, 2 capas ocultas y 2 variables de salida. Extraído de [15].

Crear modelos con RNA es un proceso empírico, es decir, la cantidad de capas y neuronas del modelo es un proceso de prueba y error, con el objetivo de minimizar el error y mejorar la capacidad de la red para realizar predicciones. Para esto, se debe realizar una búsqueda para encontrar la mejor combinación.

- **Random Forest:** es un método de aprendizaje que combina múltiples árboles de decisión individuales. Los árboles de decisiones consisten en nodos y hojas, cada nodo representa una decisión basada en el valor de una característica, y cada hoja representa una predicción [15].

  Utiliza el método *Ensemble Learning*, que se basa en la idea de que la combinación de múltiples modelos forma un modelo más robusto que cualquier modelo individual por separado. Además, utiliza la técnica *bootstrapping,* que consiste en crear múltiples árboles de decisión donde cada uno es entrenado a partir de diferentes subconjuntos de datos, y luego combinar estos modelos para mejorar la estabilidad y precisión de las predicciones [12].

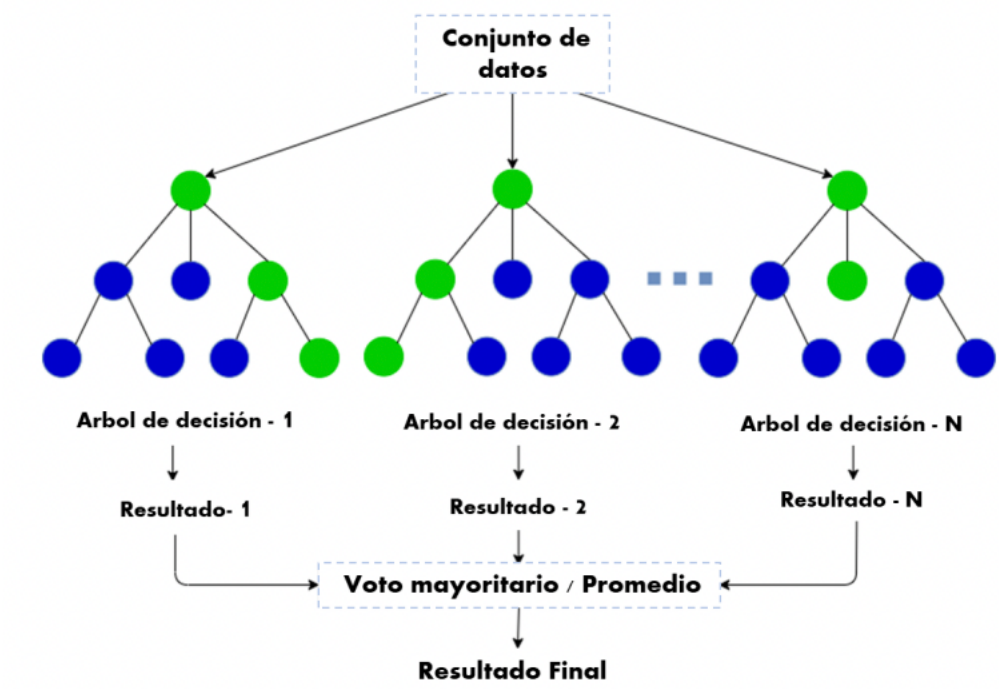


Figura 12. Representación de regresión con Random Forest. Extraído de [16].

En el caso de problemas de regresión, para generar la predicción final se utiliza el promedio de las predicciones de todos los árboles.

En Random Forest no se produce overfitting por un exceso de árboles, ya que solo puede hacer que mejore el resultado. Sin embargo, añadir árboles una vez que el modelo se estabiliza es una pérdida de recursos computacionales.

Al construir los modelos predictivos se busca encontrar una serie de parámetros que refiere a los coeficientes numéricos propios de cada modelo y que se obtiene de forma automática durante el proceso de entrenamiento. Por el contrario, los hiperparámetros corresponden a coeficientes externos al modelo, es decir, no se obtienen automáticamente y deben ser definidos manualmente [14].

4.4.1 Métricas de rendimiento

Para evaluar los modelos y determinar cuál es el más adecuado para el problema planteado, es necesario medir el rendimiento y precisión de cada modelo. Para ello, se utilizan métricas de

rendimiento que identifican la tasa de error entre los valores reales y los valores predichos. Las métricas empleadas corresponden a [17]:

- **Error Absoluto Medio (MAE):** media aritmética de la sumatoria de la diferencia entre el valor predicho y los valores reales.

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

Donde y_i corresponde al valor real, $\hat{y}_i$ el valor predicho, *N* el número de muestras y i el iésimo valor.

- **Error Cuadrático Medio (MSE):** se calcula como la media aritmética de las diferencias al cuadrado entre el valor predicho y los valores reales en un conjunto de datos. Tiene el efecto de castigar a los errores más grandes, es decir, cuando mayor sea la diferencia entre la predicción y el valor real, mayor será el valor del MSE.

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

Donde y_i corresponde al valor real, $\hat{y}_i$ el valor predicho, *N* el número de muestras y i el iésimo valor.

- **Raíz del Error Cuadrático Medio (RMSE):** raíz cuadrada de la media aritmética de la sumatoria de los cuadrados de los errores entre el valor predicho y los valores reales. Es una interpretación en la misma escala que las predicciones y valores reales.

$$RMSE = \sqrt{MSE}$$

- **Coeficiente de determinación (R^2):** mide la proporción de la varianza en los datos dependientes que es explicada por el modelo, se utiliza para evaluar la calidad del ajuste del modelo. Un valor cercano a 1 indica que el modelo se ajusta a los datos.

4.4.2 K – fold cross – validation

La validación corresponde al proceso de evaluar un modelo de Machine Learning en un conjunto de datos independiente del que se utilizó para entrenar el modelo. Dentro de las técnicas de validación más comunes se encuentra la *k – fold cross – validation*.

La validación cruzada es una técnica que divide los datos en k subconjuntos o *folds*, luego el modelo se entrena *k* veces, utilizando *k – 1 folds* para el entrenamiento y el *fold* restante para prueba. Es útil cuando se dispone de una cantidad limitada de datos, maximizando el uso de datos disponibles, y durante el ajuste de hiperparámetros de manera más fiable [12].

5. Metodología

El presente proyecto de memoria se desarrolla en cuatro etapas principales: recopilación de los datos necesarios, pre – procesamiento de los datos, modelamiento con herramientas de Machine Learning y evaluación de los modelos.

1. Recopilación de datos: en esta etapa se recolectan los datos necesarios desde dos fuentes principales: la plataforma PI System de CMDIC y la plataforma en línea del reómetro *'KRheo'* de KONATEC. Estos datos son esenciales para el desarrollo de un modelo predictivo confiable.

2. Pre – procesamiento de datos: esta etapa incluye la reducción de la cantidad de datos mediante la limpieza de datos atípicos y selección de las características significativas que se utilizarán en la generación de los modelos predictivos.

3. Modelado con Machine Learning: en esta fase se desarrollan y entrenan los algoritmos de Machine Learning para predecir el comportamiento del espesador de relaves HRT 4001. Se exploran diferentes técnicas de modelado.

4. Evaluación de modelos: finalmente, se evalúan los modelos generados utilizando métricas de rendimiento, y la extracción de conclusiones basadas en el modelo seleccionado.

La metodología planteada se representa esquemáticamente en la Figura 13.

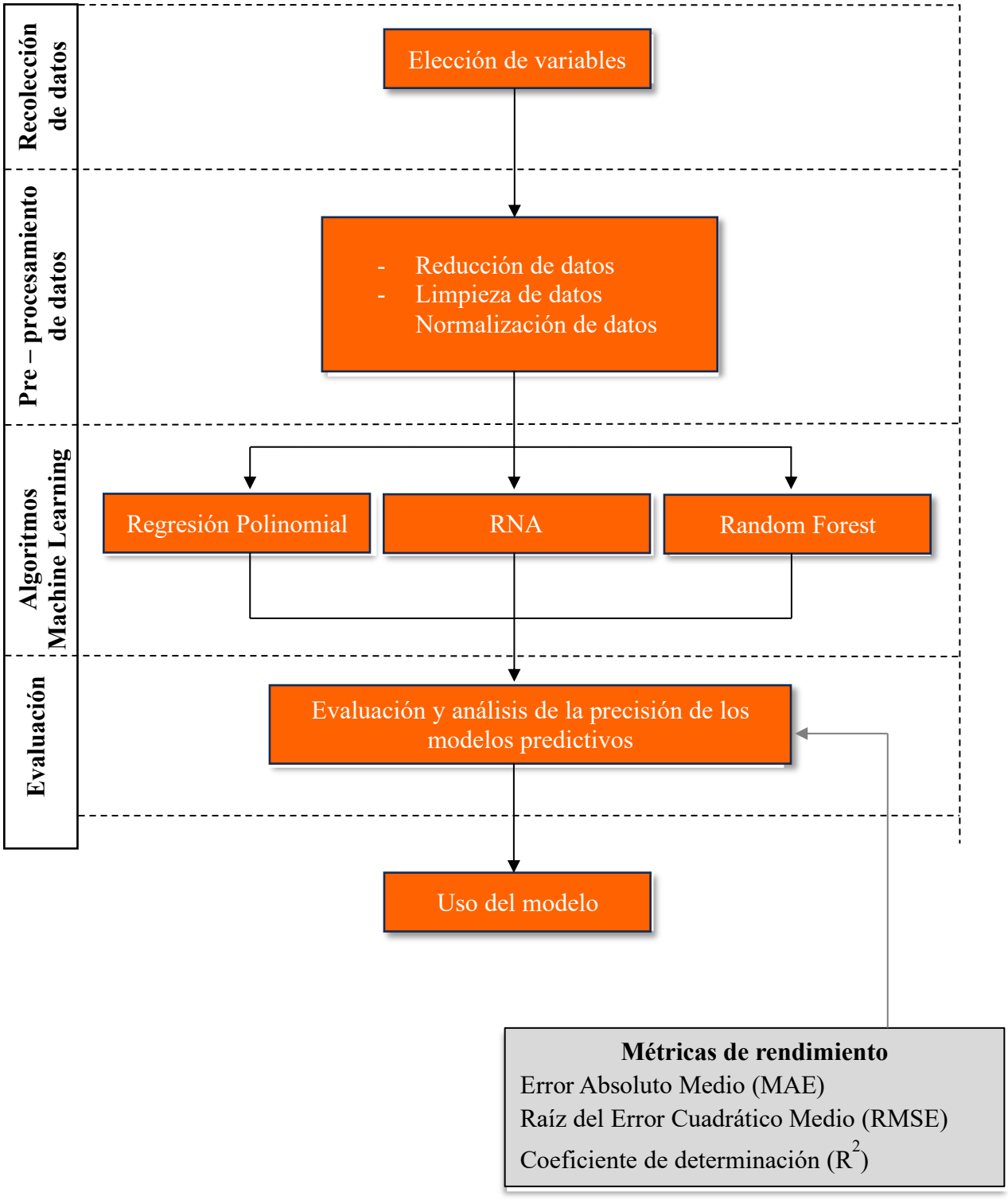


Figura 13. Diagrama de flujo de la metodología de investigación. Elaboración Propia.

6. Desarrollo Experimental

Existen diversos lenguajes de programación y plataformas para la construcción de modelos predictivos con Machine Learning. Durante este proyecto se emplea Python, considerado el lenguaje de programación universal independiente de la plataforma utilizada y el más abundante en librerías para este tipo de proyecto [18].

Las librerías utilizadas corresponden a *Numpy*, que brinda soporte para arreglos y matrices multidimensionales, *Pandas*, utilizado para análisis y estructuración de datos, como *Dataframe*, *Matplotlib y Seaborn*, para la visualización de datos, *Scikit – Learn*, que ofrece una amplia gama de algoritmos de Machine Learning, además de proporcionar herramientas para el pre – procesamiento de datos y evaluación de modelos, como las métricas de rendimiento, *TensorFlow*, que también ofrece algoritmos de Machine Learning, *Keras*, que permite implementar redes neuronales y *GridSearchCV*, para el ajuste de hiperparámetros [19].

6.1 Recolección de datos

El conjunto de datos consta de 14 variables, las cuales se dividen en 11 variables de entrada (características predictivas) y 3 variables de salida a predecir que corresponden al porcentaje de sólidos, yield stress y viscosidad de la pulpa de descarga del espesador de relave HRT 4001. Las variables por considerar en el modelamiento corresponden a:

- **Características de la pulpa de alimentación:** pH, nivel de arcillas (% caolinita, % pirofilita y % ilita) y granulometría (%-#200 *Tyler).*

- **Variables operacionales:** flujo de alimentación (tph), flujo de descarga (tph), dosis de floculante (gpt), nivel de interfase (m), nivel de cama (%), torque (%) y flujo de descarga ($m^{3/}h$).

- **Características reológicas:** yield stress (dinas/cm^2) y viscosidad (cP).

Las presentes variables corresponden a datos en línea cada 1 minuto extraídos desde *PI System* y la reología del reómetro en línea *KONATEC*, a excepción del nivel de arcillas, las cuales se encuentran inferidas cada 24 horas, por lo tanto, es necesario interpolar los datos para obtener los valores de los porcentajes cada 1 minuto.

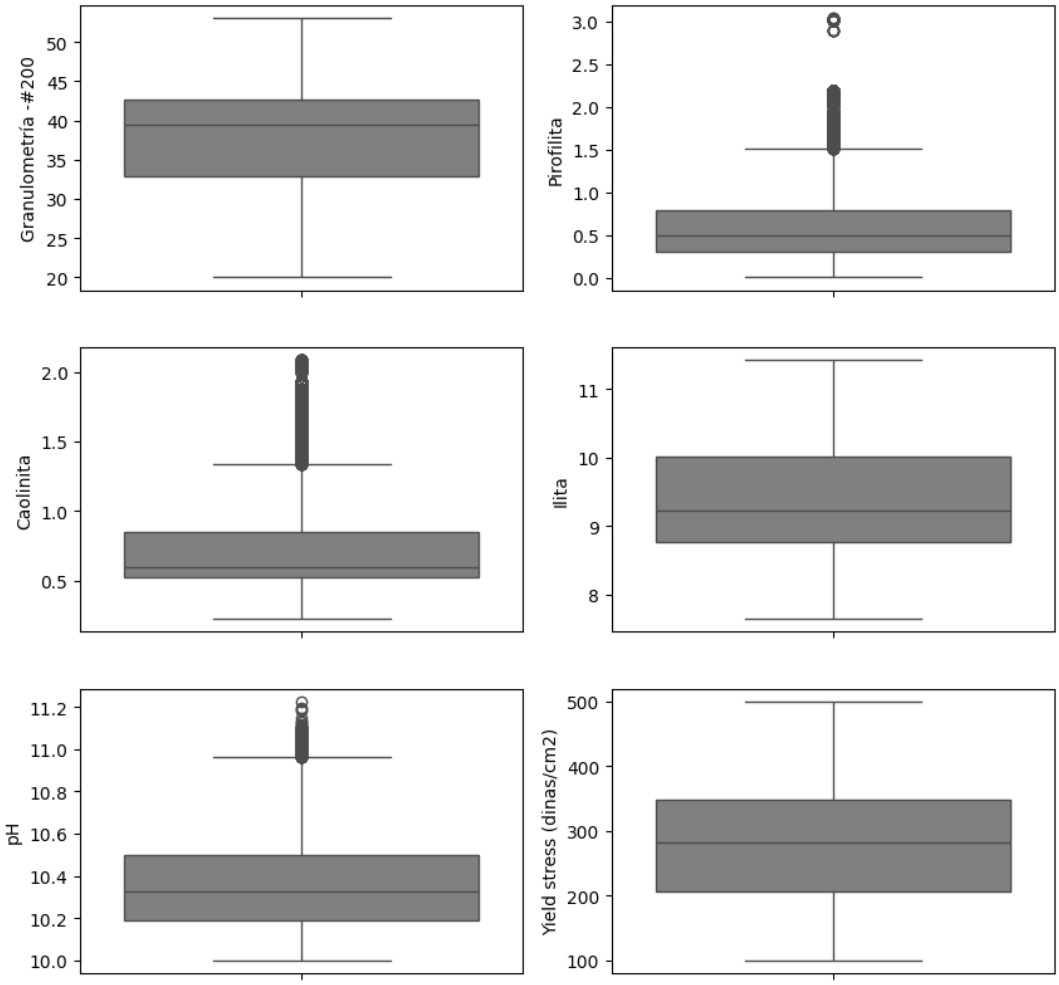
Además, los datos de la granulometría son extraídos desde los *CYCLONEtrac PST – Sistema de Rastreo de Tamaño de Partículas*, que entrega información en tiempo real del tamaño de partículas del overflow de cada hidrociclón, donde se selecciona la malla 200, es decir, 75 micrones, que corresponde al tamaño más pequeño rastreado.

Estos datos, proveniente de distintas fuentes, deben ser procesados de manera de ser transformados en información útil. Para esto se utiliza el *método ETL*, referido a *Extraction, Transformation y Loading*, que consiste en obtener la información necesaria desde las distintas fuentes, y luego de ser transformados, cargarlos en un formato más amigable. Por lo tanto, se consolidó una base de datos, concatenando toda la información en un *Dataframe*, con el que se trabaja durante todo el proyecto.

6.2 Pre – procesamiento de datos

El objetivo principal del pre – procesamiento de datos es la transformación de la data cruda en un formato limpio y adecuado para el entrenamiento y eficacia de los modelos predictivos. Esto implica verificar si existen datos incompletos, eliminar valores duplicados y valores atípicos u outliers. De no corregir estos valores se generaría un modelo que no representaría el comportamiento real del sistema a predecir.

Para identificar y visualizar los datos atípicos de cada variable es posible utilizar gráficos de dispersión y box – plots, o gráficos de caja. Estos gráficos proporcionan una visión clara de la dispersión de los datos y ayudan a identificar cualquier anormalidad que pueda afectar la calidad del modelo. Los gráficos de caja del conjunto de datos de la operación del espesador de relaves HRT 4001 se presentan a continuación.



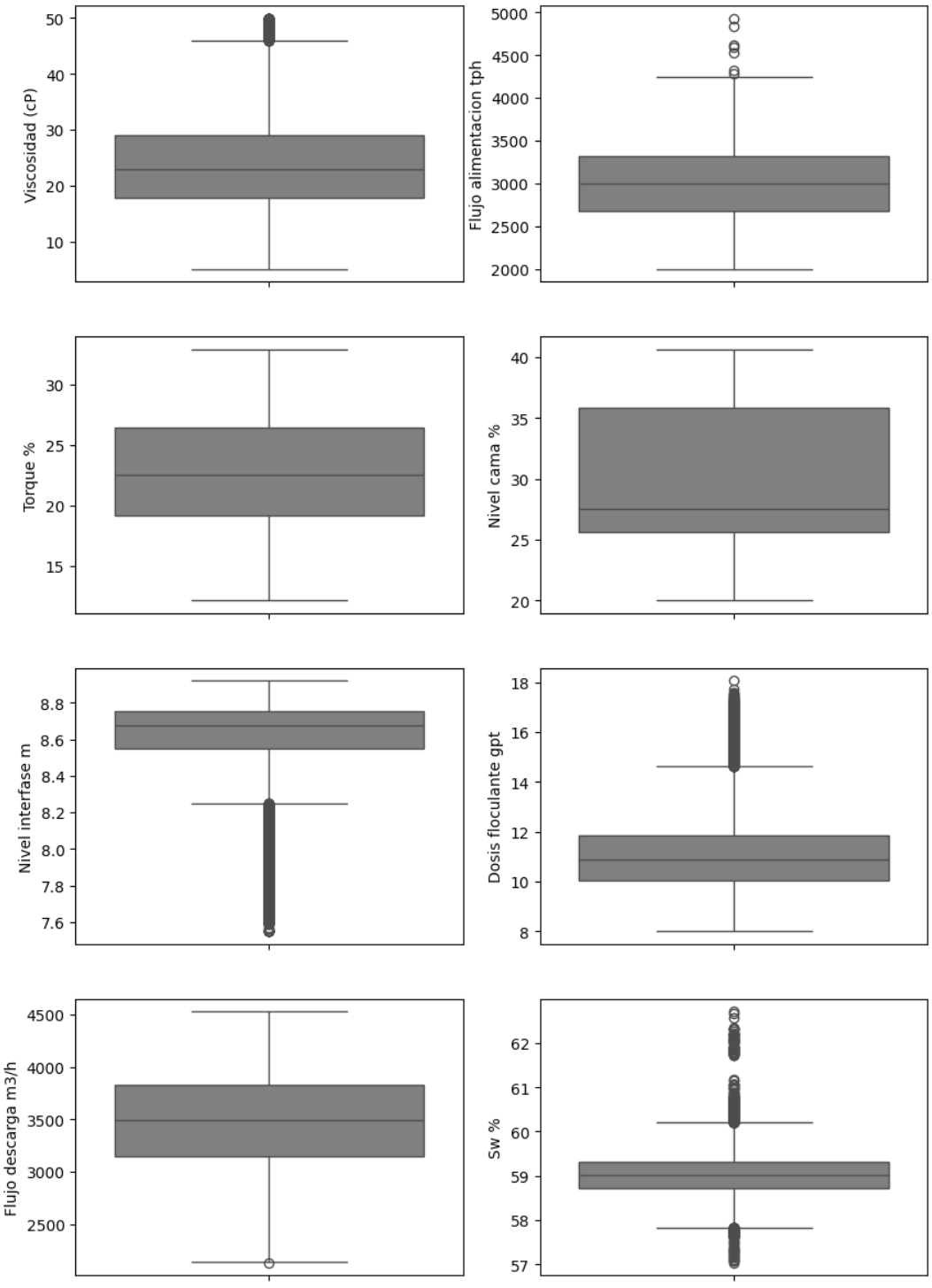


Figura 14. Gráficos de caja de data sucia de todas las variables seleccionadas para modelar el espesador de relaves HRT 4001.

Cada gráfico representa la distribución del conjunto de datos de cada variable elegida, donde se visualiza la caja desde el primer hasta el tercer cuartil, representando el rango intercuartílico, la

mediana, que corresponde a la línea horizontal central que divide el 50% central de los datos y los puntos fuera de la caja indican los valores atípicos.

Posterior a la limpieza, se exploran nuevamente los datos para saber el rango de valores que toma cada una de las variables. Para esto, se utiliza el método *describe(),* obteniendo datos estadísticos como la media, desviación estándar, valores mínimos, valores máximos, etc.

6.3 Estandarización de los datos

Muchas técnicas de Machine Learning requieren que todas las características estén en la misma escala para funcionar correctamente, para esto es necesario realizar la estandarización de los datos.

La estandarización es el proceso de transformar las características de los datos para que tengan una media de 0 y una desviación estándar de 1. Este procedimiento garantiza que todas las características contribuyan por igual en el entrenamiento del modelo, asegurando que ninguna característica domine a las demás debido a su diferencia de escala y distribución, como se aprecia en la Figura 15.

Se utiliza el transformador *MinMaxScaler()* de la librería *Scikit – Learn,* que ajusta cada característica al intervalo [0,1]. La fórmula para escalar una variable x usando este transformador es:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Donde x' corresponde a la variable escalada, x es el valor original de la variable, x_{min} es el valor mínimo de la variable y x_{max}es el valor máximo [12].

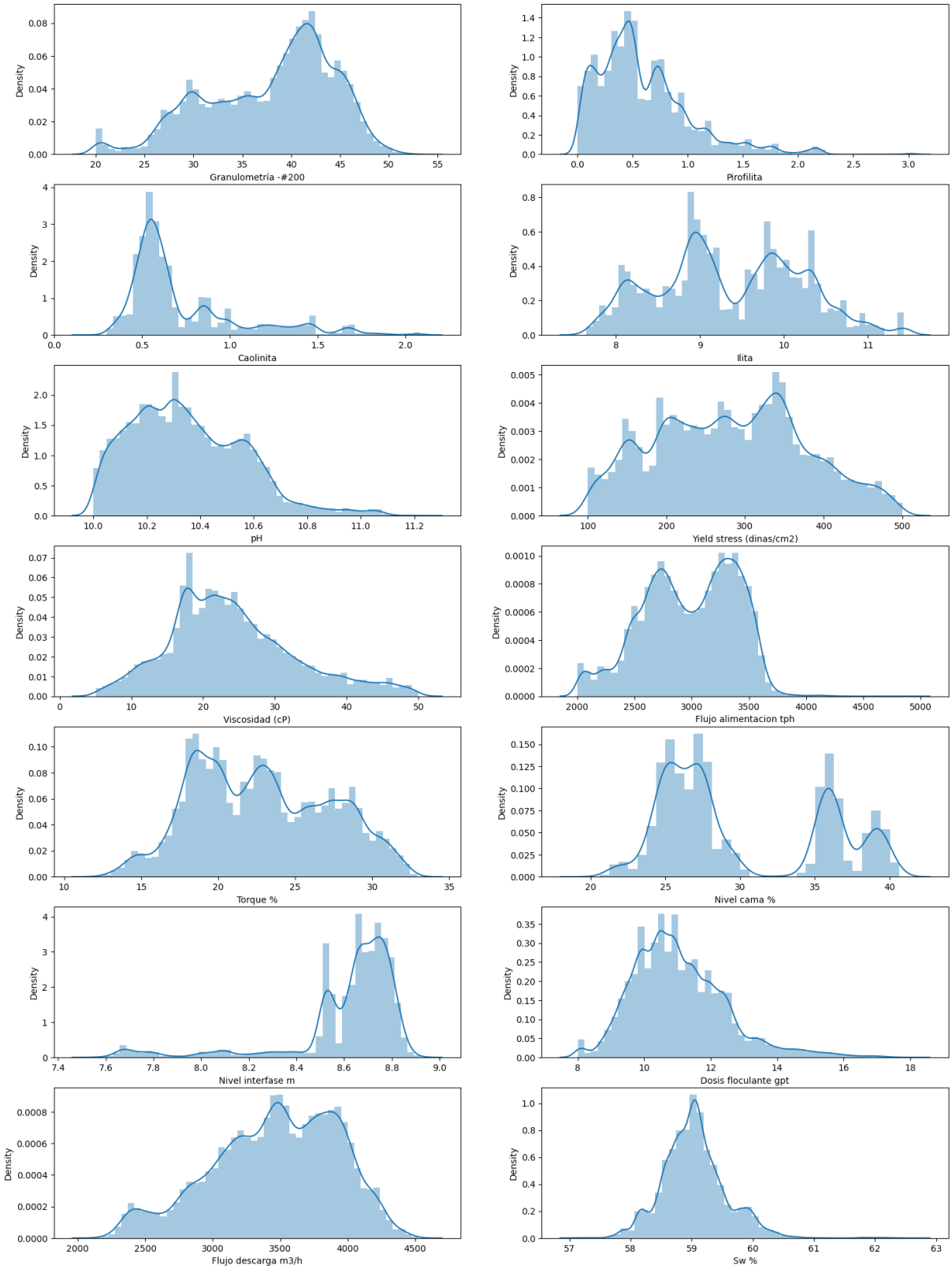


Figura 15. Distribución de todas las variables involucradas, posterior a la limpieza, en el modelamiento del espesador HRT 4001.

6.4 División datos en conjunto de entrenamiento y prueba

Para evaluar el rendimiento de los modelos de manera objetiva y evitar problemas de overfitting, los datos se dividen en un conjunto de entrenamiento y en un conjunto de prueba, que corresponde al 70% y 30% del conjunto de datos original divididos aleatoriamente. Los datos del conjunto de entrenamiento se catalogan como *X_train* e *y_train*, mientras que el conjunto de prueba se cataloga como *X_test* e *y_train*, utilizando la herramienta *train_test_split*.

El conjunto de entrenamiento es utilizado para entrenar el modelo, es decir, para que el modelo aprenda las relaciones entre las variables predictoras y objetivo. En cambio, el conjunto de prueba se reserva para evaluar el rendimiento del modelo.

6.5 Modelado y entrenamiento de algoritmos

En este paso se entrenan los modelos con los algoritmos seleccionados y los datos de entrenamiento, con el objetivo de identificar y comparar el desempeño y efectividad de cada uno respecto a la base de datos otorgada.

- **Modelo con Regresión Polinomial Multivariable:** se define en Python una función para crear y entrenar un modelo de regresión polinómica y luego se utiliza esta función para entrenar un modelo para cada variable objetivo.

  Los parámetros de la función corresponden a *X_train, y_train y degree*. El primer parámetro corresponde al conjunto de datos de entrenamiento de las características predictoras, el segundo parámetro corresponde al conjunto de entrenamiento para las variables objetivos y *degree* corresponde al grado del polinomio.

  Se utiliza la herramienta *'PolynomialFeatures'* de la biblioteca *sklearn.preprocessing*, para transformar las características originales en características polinómicas.

  Finalmente, se entrenan múltiples modelos con regresión polinómica, uno para cada variable objetivo, utilizando la misma función de entrenamiento.

- **Modelo con Redes Neuronales Artificiales:** se implementan modelos RNA para predecir múltiples variables objetivo. Para la creación de la red se utilizan las librerías *Tensorflow* y *Keras*.

  Los datos se cargan desde un archivo Excel, además, se calculan los valores máximo y mínimos de cada columna para su uso en el escalado.

  Se crea un modelo secuencial agregando capas de una en una hasta que se complete la topología de la red. Para esto, se utilizan las herramientas *'Sequential'* para construir la red en el que las capas se añaden de manera secuencial y lineal, y *'Dense'* para ir añadiendo las capas, especificando el número de neuronas y la función de activación para cada capa, asegurando que cada neurona está conectada a todas las neuronas de la capa siguiente, permitiendo la combinación de características.

  Se especifica que la capa de entrada tenga el número correcto de variable de entradas, utilizando el argumento 'imput_dim'. Además, la función de activación a utilizar corresponde a ReLu, debido a que evita el problema del gradiente de desvanecimiento a diferencia de las otras funciones de activación. El fenómeno gradiente de desvanecimiento impide que las capas aprendan adecuadamente, ya que los gradientes se vuelven extremadamente pequeños y no contribuyen a la actualización de los pesos.

  En la capa de salida, que posee 3 neuronas, correspondientes a la cantidad de variables a predecir, no se utiliza ninguna función de activación, que es la recomendación para los problemas de regresión.

  Ya definido el modelo, se debe compilar. Para esto, se especifican características adicionales requeridas para entrenar el modelo, como el optimizador y la función de pérdida. La función de pérdida es la métrica que mide la diferencia entre las predicciones y los valores reales, y es la función matemática que el modelo intenta minimizar durante el entrenamiento, en este caso se utiliza el MSE. El optimizador ajusta los pesos de la red para minimizar la función de pérdida durante el entrenamiento, en este caso se utiliza el optimizador *'adam'.*

Luego, para entrenar los modelos, se deben definir dos parámetros. El primero corresponde a los *'epochs'*, que corresponde al número de veces que el algoritmo de entrenamiento pasará por todo el conjunto de datos de entrenamiento, y el *'batch_size'*, que refiere al número de instancias que se evalúan antes de que se realice una actualización de los pesos en la red.

Finalmente, se evalúa el modelo con los datos de prueba y se generan predicciones, que deben ser reescaladas a su escala original utilizando los valores máximos y mínimos calculados previamente.

- **Modelo con Random Forest:** para la creación del Random Forest se utiliza la herramienta *'RandomForestRegressor'* importada de la librería *sklearn.ensemble*.

  De igual forma que los modelos anteriores, los datos se cargan desde un archivo Excel y se calculan los valores mínimos y máximos de cada columna para escalar.

  Para entrenar el modelo deben ser especificados el número de árboles del Random Forest, la profundidad máxima de cada árbol y el número máximo de características a considerar para dividir un nodo en cada árbol.

  Finalmente, se evalúa el modelo y se generan predicciones, reescalando a su escala original.

Para evaluar todos los modelos se utiliza la librería *sklearn.metrics*, que posee las métricas mencionadas a utilizar. Con este paso se verifica la capacidad predictiva del modelo.

6.6 Ajuste de hiperparámetros con GridSearchCV()

El ajuste de hiperparámetros es una fase crucial en el desarrollo de modelos predictivos. *GridSearchCV* es una técnica que permite realizar una búsqueda exhaustiva sobre múltiples configuraciones de un conjunto de posibles valores de hiperparámetros para encontrar la combinación óptima que maximiza el rendimiento [19]. Esta técnica forma parte de la biblioteca *scikit – learn* de Python. Si bien corresponde a una herramienta poderosa, se debe tener en consideración la gran exigencia computacional que genera.

El conjunto de posibles valores para los hiperparámetros se presenta a continuación.

Tabla 2. Conjunto de posibles valores de hiperparámetros para las diferentes técnicas de regresión.

Modelo	Conjunto de posibles valores
Regresión Polinomial Multivariable	*'poly_degree':* [2, 3, 4] *'regressor_fit_intercept':* [True, False]
RNA	*'hidden_layers':* [1, 2, 3] *'neurons':* [(32, 32, 32), (64, 64, 64), (128, 128, 128), (32, 64, 128), (64, 128, 256), (128, 64, 32)] *'batch_size':* [50, 100, 200, 300, 400, 500, 600] *'epochs':* [500, 600, 700, 800, 900, 100]
Random Forest	'*n_estimators':* [400, 500, 600, 700, 800, 1000, 1500] '*max_depth':* [None, 10, 20] '*max_features':* ['auto', 'sqrt', 'log2']

Esta técnica utiliza validación cruzada interna para cada combinación de hiperparámetros. En este caso se utilizaron 5 subconjuntos para cada modelo, en cada iteración el modelo se entrena en alguno de estos subconjuntos y se valida en los conjuntos restantes, proporcionando una estimación robusta del rendimiento del modelo de cada posible combinación, seleccionando la combinación de hiperparámetros que da el mejor rendimiento a través de esos subconjuntos.

Es por esto que, es necesario entrenar múltiples modelos con diferentes parámetros e hiperparámetros para posteriormente elegir cual es el modelo más adecuado para el problema a resolver luego del ajuste de dichos coeficientes.

En la Regresión Polinomial Multivariable, el hiperparámetro clave está relacionado con la estructura del modelo, es decir, el grado del polinomio que se va a ajustar a los datos, este hiperparámetro controla cuántos términos polinomiales de cada variables y combinaciones entre variables se incluyen en el modelo.

En el caso de las RNA, para obtener los parámetros más adecuados se debe definir, por ejemplo, el número de iteraciones del algoritmo de entrenamiento o el número de capas ocultas de la red neuronal, los que corresponden a los hiperparámetros del modelo ya que son definidos al momento de crear la red.

En Random Forest, el número de árboles no es un hiperparámetro crítico ya que, añadir árboles solo puede mejorar el resultado, sin embargo, añadir más árboles una vez que la mejora se estabiliza es una pérdida de recursos computacionales.

7. Resultados

Luego del pre – procesamiento de los datos, se realizó un análisis estadístico para verificar la limpieza. Este análisis incluyó la evaluación de valores de interés como el mínimo, máximo, promedio y desviación estándar, donde se observó que las variables con los datos limpios se encuentran dentro de los rangos operacionales normales del espesador de relaves HRT 4001, permitiendo utilizar estos datos para modelar los algoritmos de Machine Learning. Los resultados se presentan a continuación en la Tabla 3.

Tabla 3. Resultado de análisis estadístico básico de la limpieza de datos.

	Mínimo	Máximo	Promedio	Desviación estándar
Flujo alimentación, tph	2.000	5.032	2.978	400
pH	10,0	11,2	10,4	0,2
Yield stress, dinas/cm^2	100	499	282	94,
Viscosidad, cP	5,1	50	24,2	8,9
Torque, %	12	52	23	2
Nivel cama, %	20	41	30	2,6
Nivel interfase, m	6,6	8,9	8,6	0,2
Dosis floculante, gpt	8,0	18,1	11,1	1,4
Flujo descarga, m3/h	2.132	4.525	3.458	464
Granulometría malla 200, %	20	53	38	6,5
Pirofilita, %	0,01	3,04	0,60	0,42
Ilita, %	7,65	11,42	9,35	0,83
Caolinita, %	0,22	2,08	0,73	0,33
Porcentaje sólidos descarga, %	55,9	62,7	59,1	0,51

Los valores obtenidos del ajuste de hiperparámetros que obtienen la mejor precisión y rendimiento de los modelos, luego de entrenarse y validarse cinco veces, se presentan en la Tabla 4.

Tabla 4. Resultado de los mejores valores ajustados de hiperparámetros para las diferentes técnicas.

Modelo	Valores de hiperparámetros
Regresión Polinomial Multivariable	*'poly_degree':* 4, *'regressor_fit_intercept':* True
RNA	*'hidden_layers':* 3, *'neurons':* (64, 128, 256), *batch_size':* 300, *'epochs':* 900
Random Forest	'*n_estimators':* 800, '*max_depth':* None, *max_features':* sqrt

Se grafican las curvas de las métricas de rendimiento MAE y MSE de la RNA creada, que permite observar cómo evolucionan las métricas de rendimiento a medida que progresa el entrenamiento de la red neuronal artificial. Estas curvas se presentan en la Figura 16.

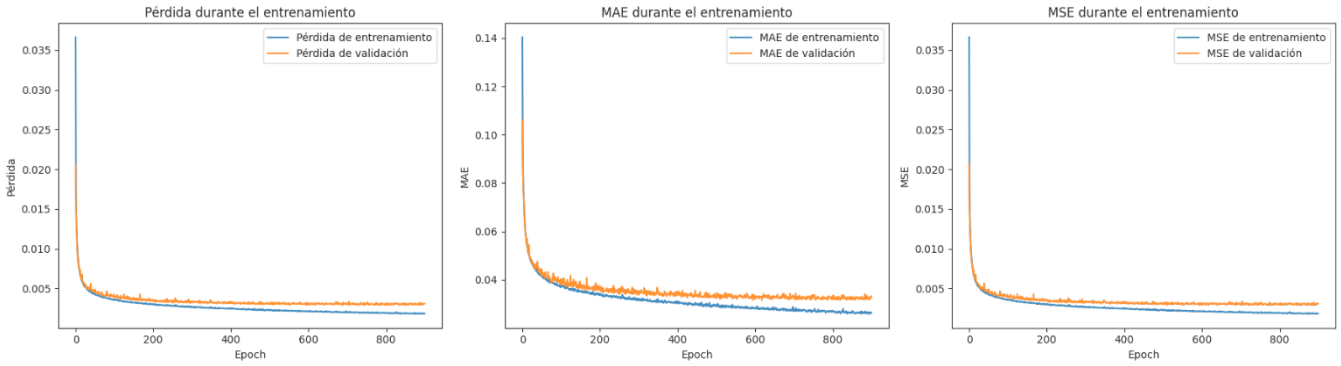


Figura 16. Curvas de métricas de rendimiento MAE y MSE de la red neuronal creada.

La línea azul representa la forma en que evoluciona cada métrica en el conjunto de datos de entrenamiento a lo largo de las iteraciones. La línea naranja muestra cómo evoluciona cada métrica en el conjunto de datos de validación, es decir, en los datos no utilizados durante el entrenamiento para evaluar la capacidad del modelo.

Además, se observa que, ambas curvas disminuyen juntas de manera similar, lo que indica que el modelo RNA está aprendiendo de manera consistente. Sin embargo, al final de la curva una leve separación, lo que puede indicar un ligero overfitting, pero es aceptable ya que la diferencia no es significativa. Además, la separación indica que el modelo tiene un mejor rendimiento en los datos de entrenamiento comparado con los de validación, lo cual es el comportamiento esperado ya que el modelo se entrena directamente con el primer conjunto.

En la Figura 17 se presentan las métricas de rendimiento del mejor modelo encontrado para cada algoritmo, es decir, el que obtuvo la mejor combinación de hiperparámetros. Para evaluar el rendimiento, con respecto a cada variable objetivo, se utiliza el conjunto de prueba que no fue utilizado en la validación cruzada interna en la búsqueda de los hiperparámetros.

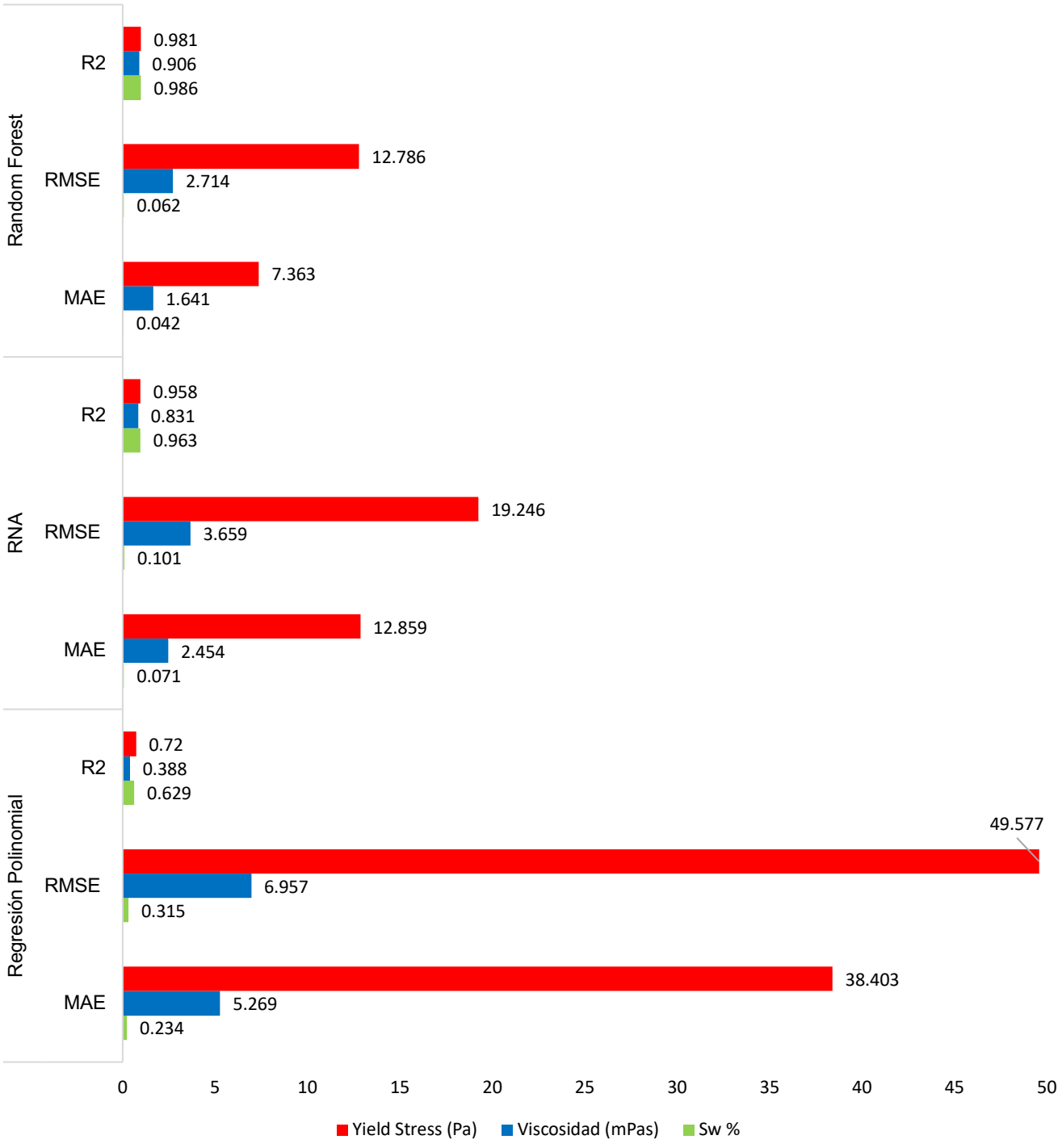


Figura 17. Visualización valores de las métricas de rendimientos de los modelos predictivos creados con Machine Learning respecto a cada variable objetivo.

Al evaluar los modelos predictivos desarrollados para predecir el comportamiento del espesador de relaves HRT 4001 para las variables yield stress, viscosidad y porcentaje de sólidos

en la pulpa de descarga, se observa que el modelo con Random Forest presenta los menores errores, tanto en términos de MAE como RMSE para las tres variables objetivo, indicando que las predicciones están cerca de los valores reales y sugiere que el modelo maneja mejor los errores grandes, siendo más robusto frente a grandes desviaciones.

Además, obtiene los mayores valores de R^2, demostrando que el modelo se ajusta mejor frente a la RNA y la regresión polinomial. Eso lo hace más confiable para ser utilizado para predecir el comportamiento del espesador de relave.

Para corroborar que no existe sobreajuste en los modelos entrenados es necesario re – evaluarlos, para esto se realiza validación cruzada comparando los RMSE de cada modelo por separado. Esto permite evaluar el rendimiento de los modelos de forma más robusta al promediar los valores de los RMSE obtenidos en cada iteración. Los valores obtenidos se presentan en la siguiente tabla.

Tabla 5. Comparación entre RMSE de cada modelo con RMSE con validación cruzada.

Modelo	**Métrica**	**Variables a predecir**		
		Sw descarga %	Viscosidad (cP)	Yield stress (dinas/cm^2)
Regresión Polinomial Multivariable	RMSE	0,315	6,957	49,5
	RMSE CV	0,40	7,231	50,3
RNA	RMSE	0,101	3,659	19,246
	RMSE CV	0,120	3,864	19,974
Random Forest	RMSE	0,062	2,714	12,786
	RMSE CV	0,077	2,950	13,236

Se observa que los valores de RMSE entre validación cruzada y el conjunto de prueba de cada modelo, para cada variable objetivo, son similares. Por lo tanto, se descarta un sobreajuste, además, se corrobora que el mejor modelos corresponde al Random Forest.

Las Figuras 18, 19 y 20 muestran los gráficos de los valores reales versus los valores predichos por cada modelo. De izquierda a derecha considera el porcentaje de sólidos en la pulpa de descarga (Sw %), la viscosidad (cP) y el yield stress (dinas/cm^2), respectivamente.

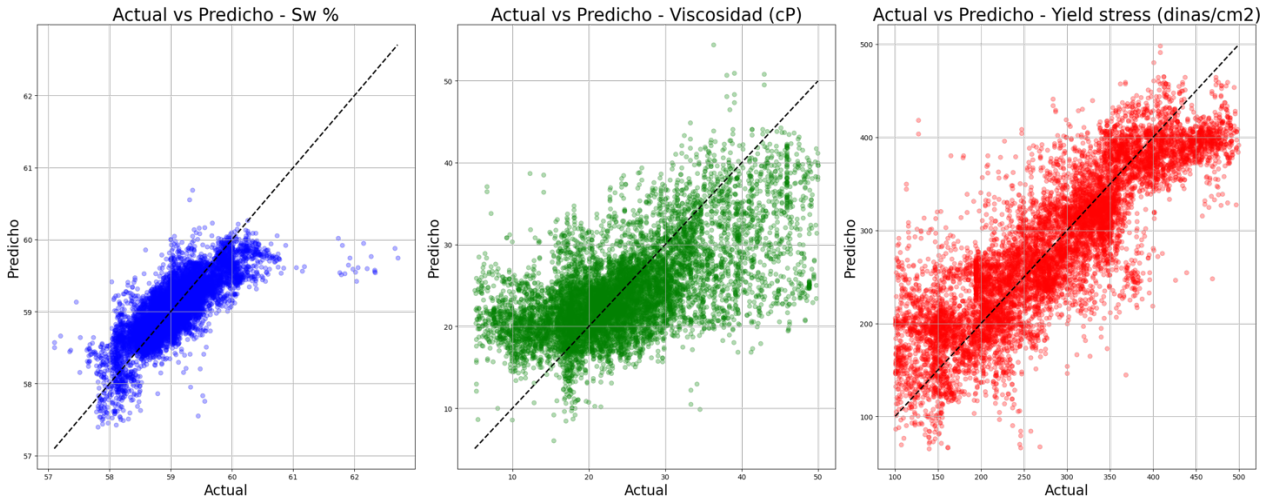


Figura 18. Gráficos de valores reales versus valores predichos con regresión polinomial multivariable.

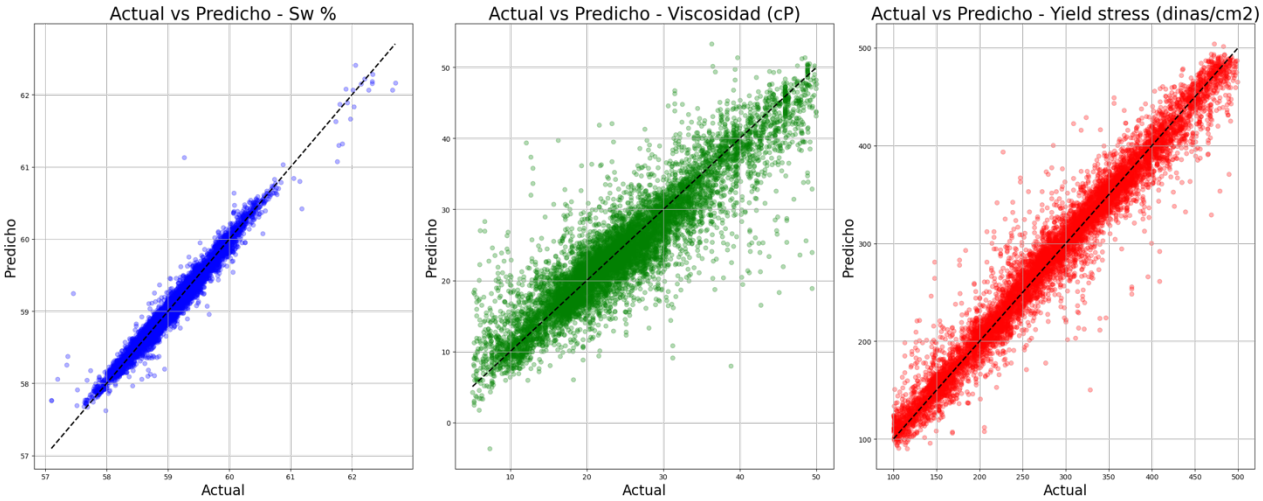


Figura 19. Gráficos de valores reales versus valores predichos con redes neuronales artificiales.

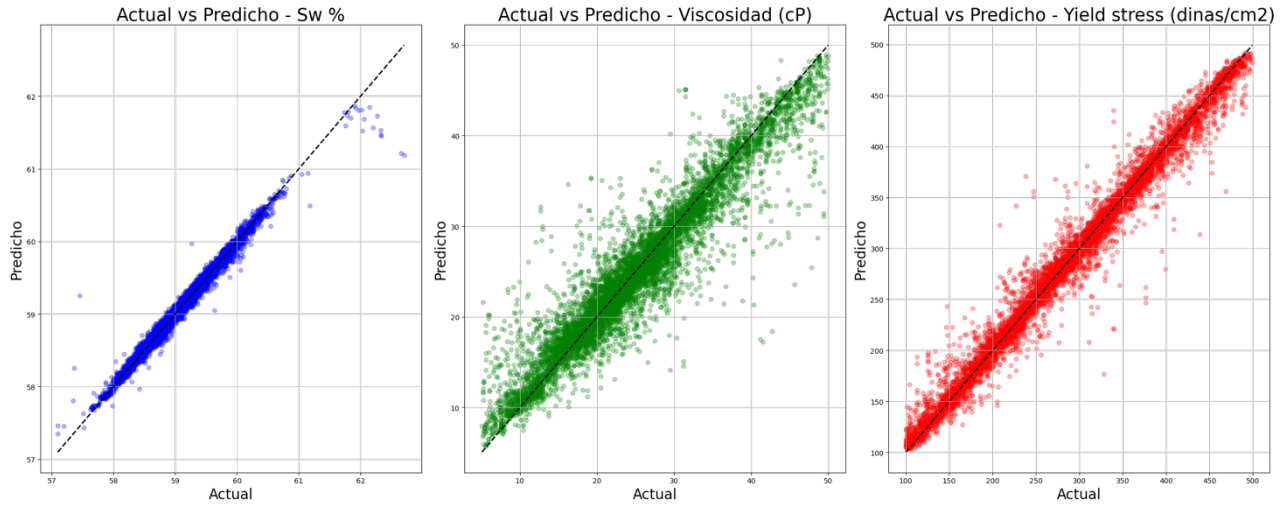


Figura 20. Gráficos de valores reales versus valores predichos con random forest.

En los gráficos presentados, la línea diagonal representa el escenario ideal de un modelo perfecto, donde las predicciones coinciden perfectamente con los valores reales. La dispersión de los puntos alrededor de esta línea indica la precisión del modelo con respecto a cada variable objetivo.

Además, se visualiza que el modelo RNA tiene una buena capacidad para realizar predicciones cercanas, tal como indicaban los valores de las métricas de rendimiento y las curvas de evolución del entrenamiento, pero es superado por el Random Forest, que presenta predicciones más precisas.

Considerando el algoritmo con mejor rendimiento, se puede evaluar la importancia de los parámetros de entrada, calculada mediante el incremento del error tras la permutación, presentado en la Figura 21.

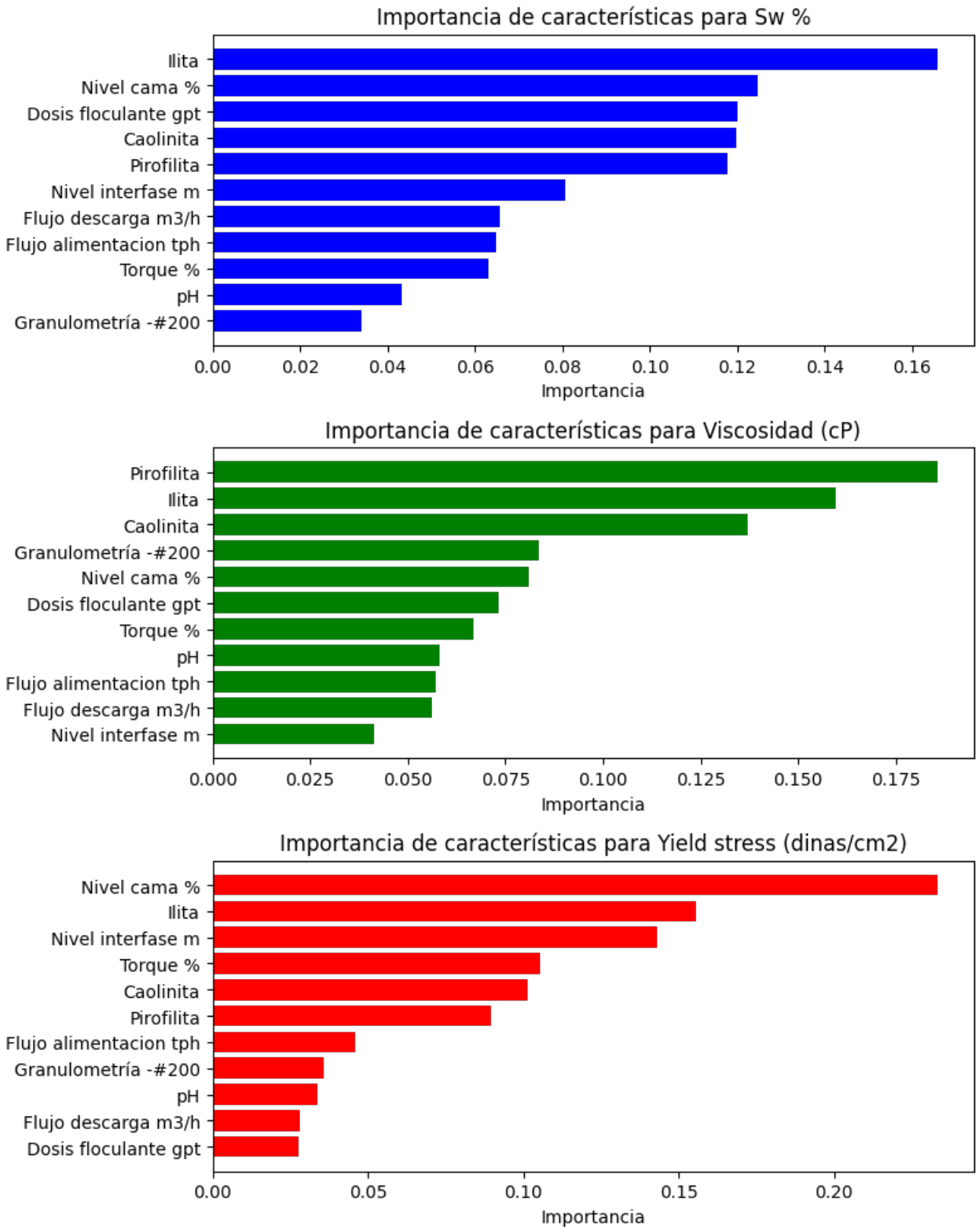


Figura 21. Importancia de las variables predictoras para cada objetivo con el modelo Random Forest.

Si bien esta herramienta es computacionalmente costosa, ya que implica entrenar y evaluar el modelo varias veces más, es fácil de implementar y entrega información útil para el análisis de datos.

Esta técnica evalúa la importancia de las características de manera independiente para cada objetivo utilizando un solo modelo entrenado con las mismas características, midiendo el impacto de la eliminación aleatoria de cada variable en el rendimiento del modelo, cuanto mayor sea la disminución del rendimiento del modelo después de eliminar una característica, mayor será su importancia. De ahí la necesidad de evaluar los parámetros con el modelo que posee mejor rendimiento.

Por lo tanto, al observar la ilustración anterior, se puede notar que las arcillas, principalmente la ilita, poseen la mayor relevancia, disminuyendo el porcentaje de sólidos en la pulpa de descarga y aumentando el yield stress y viscosidad. Este hallazgo es coherente con el comportamiento esperado de las arcillas en el agua, debido a que influyen en la sedimentación y floculación en el tratamiento de relaves, en especial la ilita, que tiene propiedades hinchables, ocasionando los problema mencionados anteriormente.

Aunque las características de la pulpa de alimentación, como la composición de arcillas, no pueden ser modificadas al llegar a la planta concentradora, el modelo permite experimentar con las variables operacionales. Esto facilita identificar que ajustes en las variables operacionales se deben realizar para alcanzar el porcentaje de sólidos deseado en la pulpa de descarga, informando al mismo tiempo su reología.

8. Conclusiones

El presente proyecto de memoria ha validado la efectividad de modelos predictivos basados en técnicas de Machine Learning para predecir el comportamiento del espesador de relaves HRT 4001 de CMDIC.

Se desarrollaron y compararon tres modelos de regresión: regresión polinomial multivariable, redes neuronales artificiales y Random Forest. Tras analizar los resultados, el modelo Random Forest destacó como el más preciso y confiable, superando a los otros dos enfoques en robustez.

La elección del modelo Random Forest se justifica por el análisis de las métricas de rendimiento seleccionadas, demostrando una notable capacidad para manejar la no linealidad de las variables utilizadas y su resistencia al sobreajuste gracias a la combinación de 800 árboles de decisión. El sobreajuste ocurre cuando un modelo se ajusta demasiado bien a los datos de entrenamiento, lo que resulta en un mal desempeño con nuevos datos no vistos. El Random Forest, mediante la agregación de múltiples árboles de decisión, logra generalizar mejor y evita este problema.

Finalmente, se sugiere la ampliación del conjunto de datos a lo largo del tiempo, lo que permitiría mejorar la precisión y robustez del modelo predictivo, especialmente por los cambios futuros de las características de la pulpa de alimentación, además de la revisión de los sensores del sistema de rastreo del tamaño de partículas, ya que los datos no se encuentran disponibles de manera continua y fomentar la cultura de toma de decisiones basada en la ciencia de datos.

9. Bibliografía

[1] COCHILCO, 2023. Proyección de demanda de agua en la minería del cobre 2023 – 2034. *Comisión Chilena del Cobre.*

[2] COCHILCO, 2023. Agua en la minería del cobre. Actualización al año 2022. *Comisión Chile del Cobre*.

[3] Compañía Minera Doña Inés de Collahuasi. (2024). *Reporte de sustentabilidad 2023*.

[4] Servicio Nacional de Geología y Minería. *Depósito de Relaves*.

[5] Concha, F. (2014). *Solid – Liquid Separation in Mining Industry.* Springer.

[6] Concha, F. (2001). *Manual de filtración y separación*. Centro de tecnología mineral, CETTEM. Universidad de Concepción.

[7] Gómez, P. (2012). *Evaluación de pruebas de espesamiento de relaves en planta piloto.* (Proyecto para optar al título de Ingeniero Civil Químico). Pontificia Universidad Católica de Valparaíso.

[8] IMetChile. (s.f.). *Arcillas: problemas y soluciones en la operación de espesadores de relaves* [Diapositivas de PowerPoint].

[9] Muñoz, C. (2019). *Aplicación y comparación de sistemas de control PI y fuzzy en espesador piloto.* (Informe de Memoria de Título para optar al Título de Ingeniero Civil Metalúrgico). Universidad de Concepción.

[10] Acevedo, H. (2011). *Evaluación y mejora de prácticas de control operacional en espesadores de concentrado y relave*. (Informe de Memoria de Título para optar al Título de Ingeniero Civil Metalúrgico). Universidad de Concepción.

[11] Gutiérrez, L. (Semestre 1 – 2024). *Reología de suspensiones: Introducción a la reología de suspensiones* [Diapositivas de PowerPoint].

[12] Gerón, A. (2022). *Hands – On Machine Learning with Scikit – Learn, Keras & TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems*. O' Reilly Media, Inc.

[13] Molina, M. (2022). *Ensambles dinámicos de modelos Machine Learning para regresión*. (Trabajo de fin de grado en Ingeniería Informática). Escuela Politécnica Superior de Jaén. Jaén, España.

[14] Bobadilla, J. (2020). *Machine Learning y Deep Learning. Usando Python, Scikit y Keras*. (Ediciones de la U). Colombia.

[15] Ahmad, M., Mourshed, M., Rezgui, Y. (2017). Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption. Energy and Buildings, 147, 77 – 89.

[16] Rojas, K. (2022). Ciencia de Datos para Ciencias Naturales. https://bookdown.org/keilor_rojas/CienciaDatos/

[17] Castillo, A. O. (2022). *Desarrollo de modelos predictivos de regresión en la industria minera mediante el uso de algoritmo de machine Learning*. (Tesis para optar al Título Profesional de Ingeniero de Minas). Universidad Nacional Mayor de San Marcos. Lima, Perú.

[18] Rodríguez, D. (2023). *Análisis de datos y Machine Learning.* (Trabajo de fin de grado en Estadística). Universidad de Salamanca. Salamanca, España.

[19] Amat, J. (2024). Random Forest con Python. https://cienciadedatos.net/documentos/py08_random_forest_python

Anexos

Anexo I. Interpolación de arcillas

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline

# Dirección del Excel con los datos
datos = pd.read_excel('/Users/nathaliaoliverabascur/Desktop/DATOS TK-4001 EN
EXCEL/ARCILLAS.xlsx')

datos.rename(columns = {'Unnamed: 0': 'Fecha'}, inplace=True)

# Columna B del Excel
pirofilita = np.array(datos["PIROFILITA"])
# Columna B del Excel
ilita = np.array(datos["ILITA"])
# Columna B del Excel
caolinita = np.array(datos["CAOLINITA"])

# Transformación de fechas a datetime
datos['Fecha'] = pd.to_datetime(datos['Fecha'])
fechas = np.array(datos['Fecha'])
datos.set_index('Fecha', inplace=True)

# Se generan los minutos de cada fecha
fecha_minutos = pd.date_range(start=datos.index.min(),
                              end = datos.index.max(), freq='T')

# Interpolación con CubicSpline
curva_pirofilita = CubicSpline(fechas, pirofilita)
curva_ilita = CubicSpline(fechas, ilita)
curva_caolinita = CubicSpline(fechas, caolinita)

# Obtener los valores interpolados
```

```
pirofilita_interpolada = curva_pirofilita(fecha_minutos)
ilita_interpolada = curva_ilita(fecha_minutos)
caolinita_interpolada = curva_caolinita(fecha_minutos)

# Crear un DataFrame con los datos interpolados
datos_interpolados = pd.DataFrame({
    'Fecha': fecha_minutos,
    'Pirofilita': pirofilita_interpolada,
    'Ilita': ilita_interpolada,
    'Caolinita': caolinita_interpolada
})

# ================================================================================
# Gráficos
plt.figure(1)
plt.plot(fecha_minutos, curva_pirofilita(fecha_minutos), label='Interpolado')
plt.scatter(fechas, pirofilita ,c="darkorange", marker="o", label='Mediciones',
            s=10)
plt.title("Interpolación pirofilita")
plt.xlabel("Fecha")
plt.ylabel('Porcentaje arcilla')
plt.legend()
plt.show()

plt.figure(2)
plt.plot(fecha_minutos, curva_ilita(fecha_minutos), label='Interpolado')
plt.scatter(fechas, ilita, c="darkorange", marker="o", label='Mediciones',s=10)
plt.title("Interpolación ilita")
plt.xlabel("Fecha")
plt.ylabel('Porcentaje arcilla')
plt.legend()
plt.show()

plt.figure(3)
plt.plot(fecha_minutos, curva_caolinita(fecha_minutos), label='Interpolado')
plt.scatter(fechas, caolinita, c="darkorange", marker="o", label='Mediciones',
            s=10)
plt.title("Interpolación caolinita")
plt.xlabel("Fecha")
plt.ylabel('Porcentaje arcilla')
```

```
plt.legend()
plt.show()

# ================================================================================
# Exportación en Excel
data = {'Fecha':fecha_minutos,'Pirofilita':curva_pirofilita(fecha_minutos),
        'Ilita':curva_ilita(fecha_minutos),
        'Caolinita':curva_caolinita(fecha_minutos)}
datos_interpolados = pd.DataFrame(data)

# Datos interpolados exportados en Excel
datos_interpolados.to_excel('/Users/nathaliaoliverabascur/Desktop/DATOS TK-4001
EN EXCEL/datos interpolados.xlsx', index=False)
```

Anexo II. Código Regresión Polinomial Multivariable

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler


# Cargar los datos
data = '/Users/nathaliaoliverabascur/Desktop/Codigos/datalimpiafinal_HRT4001.xlsx'
datos = pd.read_excel(data, index_col=0)

# Definir las características y las variables objetivo
variables = ['Granulometría -#200', 'Pirofilita', 'Caolinita', 'Ilita', 'pH',
             'Flujo alimentacion tph', 'Torque %', 'Nivel cama %', 'Nivel interfase m',
             'Dosis floculante gpt', 'Flujo descarga m3/h']
objetivos = ['Sw %', 'Viscosidad (cP)', 'Yield stress (dinas/cm2)']

# Escalar solo las variables de entrada y objetivo por separado
scaler_X = MinMaxScaler()
X_scaled = pd.DataFrame(scaler_X.fit_transform(datos[variables]), columns=variables)

scaler_y = MinMaxScaler()
y_scaled = pd.DataFrame(scaler_y.fit_transform(datos[objetivos]), columns=objetivos)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.3,
random_state=42)

# Crear un pipeline que combine la transformación polinómica con la regresión lineal
pipeline = Pipeline([
    ('poly', PolynomialFeatures()),  # Se agregará el parámetro 'degree' en GridSearch
    ('regressor', LinearRegression())
])

# Definir el Grid de Hiperparámetros
param_grid = {
    'poly__degree': [2, 3, 4],  # Probar diferentes grados de polinomios
    'regressor__fit_intercept': [True, False]  # Probar con o sin intercepto
}
# Configurar y ejecutar GridSearchCV
```

```
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error',
n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)

# Imprimir los mejores parámetros encontrados
print(f"Best parameters: {grid_search.best_params_}")

# Evaluar el mejor modelo en el conjunto de prueba
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Reescalar predicciones y datos de prueba a su forma original
y_pred_original = scaler_y.inverse_transform(y_pred)
y_test_original = scaler_y.inverse_transform(y_test)

# Calcular métricas de error para cada variable de salida por separado
for i, objetivo in enumerate(objetivos):
    mse = mean_squared_error(y_test_original[:, i], y_pred_original[:, i])
    mae = mean_absolute_error(y_test_original[:, i], y_pred_original[:, i])
    r2 = r2_score(y_test_original[:, i], y_pred_original[:, i])

    print(f"\nResultados para {objetivo}:")
    print(f"  Error Cuadrático Medio (MSE): {mse}")
    print(f"  Error Absoluto Medio (MAE): {mae}")
    print(f"  Coeficiente de Determinación (R²): {r2}")

# Visualizar las predicciones con gráficos de dispersión
import matplotlib.pyplot as plt
colores = ['blue', 'green', 'red']
plt.figure(figsize=(25, 10))

for i, objetivo in enumerate(objetivos):
    plt.subplot(1, 3, i + 1)
    plt.scatter(test_original[:, i], pred_original[:, i], alpha=0.3, color=colores[i])
    plt.plot([test_original[:, i].min(), test_original[:, i].max()],
             [test_original[:, i].min(), test_original[:, i].max()], 'k--', lw=2)
    plt.title(f'Actual vs Predicho - {objetivo}', fontsize = 25)
    plt.xlabel('Actual', fontsize = 20)
    plt.ylabel('Predicho', fontsize = 20)
    plt.gcf().patch.set_alpha(0.0)
    plt.gca().patch.set_alpha(0.0)
    plt.grid(True)

plt.tight_layout()
plt.show()
```

Anexo III. Código Redes Neuronales Artificiales

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from scikeras.wrappers import KerasRegressor
from sklearn.model_selection import GridSearchCV, KFold, train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Cargar los datos
data =
'/Users/nathaliaoliverabascur/Desktop/Codigos/datalimpiafinal_HRT4001.xlsx'
datos = pd.read_excel(data, index_col=0)

# Calcular datos máximos y mínimos para escalar datos
maxs = datos.max()
mins = datos.min()
min_max = pd.DataFrame({'Mínimo': mins, 'Máximo': maxs})

# Escalar datos
scaler = MinMaxScaler()
data_esc = pd.DataFrame(scaler.fit_transform(datos), columns=datos.columns)

# Definir las características y las variables objetivo
variables = ['Granulometría -#200', 'Pirofilita', 'Caolinita', 'Ilita', 'pH',
             'Flujo alimentacion tph', 'Torque %', 'Nivel cama %',
             'Nivel interfase m', 'Dosis floculante gpt', 'Flujo descarga m3/h']
objetivos = ['Sw %', 'Viscosidad (cP)', 'Yield stress (dinas/cm2)']

# Dividir los datos en conjuntos de entrenamiento y prueba
X = data_esc[variables]
y = data_esc[objetivos]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```
# Función para crear el modelo RNA
def create_model(neurons=(128, 128, 128), activation='relu', optimizer='adam',
                 **kwargs):
    model = Sequential()
    model.add(Dense(neurons[0], input_dim=X_train.shape[1],
              activation=activation))
    model.add(Dense(neurons[1], activation=activation))
    model.add(Dense(neurons[2], activation=activation))
    model.add(Dense(3))  # Tres salidas

    model.compile(optimizer=optimizer, loss='mean_squared_error',
                  metrics=['mean_squared_error', 'mean_absolute_error'])
    return model

# Envolver el modelo con KerasRegressor
keras_regressor = KerasRegressor(model=create_model, verbose=0)

# Definir el Grid de Hiperparámetros
param_grid = {
    'model__neurons': [(64, 64, 64), (64, 128, 256), (128, 128, 128),
                       (128, 64, 32)],
    'model__batch_size': [50, 100, 200, 300, 400],
    'model__epochs': [500, 600, 800, 850, 900, 950, 1000]
}

# Configurar KFold para la validación cruzada
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Configurar y ejecutar GridSearchCV con KFold
grid = GridSearchCV(estimator=keras_regressor, param_grid=param_grid,
                    scoring='neg_mean_squared_error', n_jobs=-1, cv=kf,
                    verbose=1)

# Entrenar el modelo con GridSearchCV
entrenamiento_grid = grid.fit(X_train, y_train)
# Imprimir los mejores parámetros
mejor_parametros = entrenamiento_grid.best_params_
```

```
print(f"Mejores parámetros: {major_parametros}")

# Crear y entrenar el mejor modelo con los mejores parámetros encontrados
mejor_modelo = create_model(neurons=mejor_parametros['model__neurons'])
history = mejor_modelo.fit(X_train, y_train,
                           epochs=mejor_parametros['model__epochs'],
                           batch_size=major_parametros['model__batch_size'],
                           validation_data=(X_test, y_test), verbose=1)

# Evaluar el modelo con los datos de prueba
prediccion = mejor_modelo.predict(X_test)

# Reescalar datos a su forma original
pred_original = prediccion * (maxs[objetivos].values - mins[objetivos].values) +
mins[objetivos].values
test_original = y_test.values * (maxs[objetivos].values -
mins[objetivos].values) + mins[objetivos].values

# Calcular métricas de error para cada variable de salida por separado
mse_sw = mean_squared_error(test_original[:, 0], pred_original[:, 0])
mae_sw = mean_absolute_error(test_original[:, 0], pred_original[:, 0])
r2_sw = r2_score(test_original[:, 0], pred_original[:, 0])
sqrt_mse_sw = np.sqrt(mse_sw)

mse_viscosidad = mean_squared_error(test_original[:, 1], pred_original[:, 1])
mae_viscosidad = mean_absolute_error(test_original[:, 1], pred_original[:, 1])
r2_viscosidad = r2_score(test_original[:, 1], pred_original[:, 1])
sqrt_mse_viscosidad = np.sqrt(mse_viscosidad)

mse_yield_stress = mean_squared_error(test_original[:, 2], pred_original[:, 2])
mae_yield_stress = mean_absolute_error(test_original[:, 2], pred_original[:, 2])
r2_yield_stress = r2_score(test_original[:, 2], pred_original[:, 2])
sqrt_mse_yield_stress = np.sqrt(mse_yield_stress)
print("Sw %:")
print(f"  Error Cuadrático Medio MSE: {mse_sw}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_sw}")
print(f"  Error Absoluto Medio MAE: {mae_sw}")
```

```
print(f"  Coeficiente de Determinación R²: {r2_sw}")

print("\nViscosidad (cP):")
print(f"  Error Cuadrático Medio MSE: {mse_viscosidad}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_viscosidad}")
print(f"  Error Absoluto Medio MAE: {mae_viscosidad}")
print(f"  Coeficiente de Determinación R²: {r2_viscosidad}")

print("\nYield stress (dinas/cm2):")
print(f"  Error Cuadrático Medio MSE: {mse_yield_stress}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_yield_stress}")
print(f"  Error Absoluto Medio MAE: {mae_yield_stress}")
print(f"  Coeficiente de Determinación R²: {r2_yield_stress}")

# Verificar dimensiones
print(f"Predicciones originales dimensiones: {pred_original.shape}")
print(f"Datos de prueba originales dimensiones: {test_original.shape}")

# Visualizar las predicciones con gráficos de dispersión
colores = ['blue', 'green', 'red']
plt.figure(figsize=(25, 10))

for i, objetivo in enumerate(objetivos):
    plt.subplot(1, 3, i + 1)
    plt.scatter(test_original[:, i], pred_original[:, i], alpha=0.3,
color=colores[i])
    plt.plot([test_original[:, i].min(), test_original[:, i].max()],
             [test_original[:, i].min(), test_original[:, i].max()], 'k--',
lw=2)
    plt.title(f'Actual vs Predicho - {objetivo}')
    plt.xlabel('Actual')
    plt.ylabel('Predicho')
    plt.gcf().patch.set_alpha(0.0)
    plt.gca().patch.set_alpha(0.0)
    plt.grid(True)

plt.tight_layout()
```

```
plt.show()


# Graficar las métricas del modelo a lo largo del tiempo
plt.figure(figsize=(20, 5))

# Gráfico de la pérdida
plt.subplot(1, 3, 1)
plt.plot(history.history['loss'], label='Pérdida de entrenamiento')
plt.plot(history.history['val_loss'], label='Pérdida de validación')
plt.xlabel('Epoch')
plt.ylabel('Pérdida')
plt.title('Pérdida durante el entrenamiento')
plt.legend()

# Gráfico del error absoluto medio (MAE)
plt.subplot(1, 3, 2)
plt.plot(history.history['mean_absolute_error'], label='MAE de entrenamiento')
plt.plot(history.history['val_mean_absolute_error'], label='MAE de validación')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.title('MAE durante el entrenamiento')
plt.legend()

# Gráfico del error cuadrático medio (MSE)
plt.subplot(1, 3, 3)
plt.plot(history.history['mean_squared_error'], label='MSE de entrenamiento')
plt.plot(history.history['val_mean_squared_error'], label='MSE de validación')
plt.xlabel('Epoch')
plt.ylabel('MSE')
plt.title('MSE durante el entrenamiento')
plt.legend()

plt.gcf().patch.set_alpha(0.0)
plt.gca().patch.set_alpha(0.0)

plt.tight_layout()
plt.show()
```

Anexo IV. Código Random Forest

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV, KFold, train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import numpy as np


# Cargar los datos
data =
'/Users/nathaliaoliverabascur/Desktop/Codigos/datalimpiafinal_HRT4001.xlsx'
datos = pd.read_excel(data, index_col=0)


# Definir las características y las variables objetivo
variables = ['Granulometría -#200', 'Pirofilita', 'Caolinita', 'Ilita', 'pH',
             'Flujo alimentacion tph', 'Torque %', 'Nivel cama %',
             'Nivel interfase m',
 'Dosis floculante gpt', 'Flujo descarga m3/h']
objetivos = ['Sw %', 'Viscosidad (cP)', 'Yield stress (dinas/cm2)']

X = datos[variables]
y = datos[objetivos]

# Ajustar el escalador solo con las características
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)

# Ajustar el escalador solo con los objetivos
scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y)

# Guardar los escaladores
dump(scaler_X,
'/Users/nathaliaoliverabascur/Desktop/ALGORITMOS/escalador_X_RF_final.joblib')
dump(scaler_y,
'/Users/nathaliaoliverabascur/Desktop/ALGORITMOS/escalador_y_RF_final.joblib')
```

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,
test_size=0.3, random_state=42)

# Definir el modelo Random Forest Regressor
Modelo_RF = RandomForestRegressor(random_state=42)

# Definir el Grid de Hiperparámetros
param_grid = {
    'n_estimators': [400, 500, 600, 700, 800, 1000, 1500],
    'max_depth': [None, 10, 20],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Configurar KFold para la validación cruzada
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Configurar y ejecutar GridSearchCV con KFold
grid = GridSearchCV(estimator=modelo_RF, param_grid=param_grid,
                    scoring='neg_mean_squared_error', n_jobs=-1, cv=kf,
                    verbose=1)

# Entrenar el modelo con GridSearchCV
entrenamiento_grid = grid.fit(X_train, y_train)

# Imprimir los mejores parámetros
Mejor_parametros = entrenamiento_grid.best_params_
print(f"Mejores parámetros: {mejor_parametros}")

# Crear y entrenar el mejor modelo con los mejores parámetros encontrados
mejor_modeloRF = RandomForestRegressor(**mejor_parametros, random_state=42)
mejor_modeloRF.fit(X_train, y_train)

# Guardar el mejor modelo entrenado
dump(mejor_modeloRF,
'/Users/nathaliaoliverabascur/Desktop/ALGORITMOS/modelo_RF_final.joblib')
```

```
# Hacer predicciones
predicciones = mejor_modeloRF.predict(X_test)


# Reescalar predicciones y datos de prueba a su forma original
pred_original = scaler_y.inverse_transform(predicciones)
test_original = scaler_y.inverse_transform(y_test)

# Calcular métricas de error para cada variable de salida por separado
mse_sw = mean_squared_error(test_original[:, 0], pred_original[:, 0])
mae_sw = mean_absolute_error(test_original[:, 0], pred_original[:, 0])
r2_sw = r2_score(test_original[:, 0], pred_original[:, 0])
sqrt_mse_sw = np.sqrt(mse_sw)

mse_viscosidad = mean_squared_error(test_original[:, 1], pred_original[:, 1])
mae_viscosidad = mean_absolute_error(test_original[:, 1], pred_original[:, 1])
r2_viscosidad = r2_score(test_original[:, 1], pred_original[:, 1])
sqrt_mse_viscosidad = np.sqrt(mse_viscosidad)

mse_yield_stress = mean_squared_error(test_original[:, 2], pred_original[:, 2])
mae_yield_stress = mean_absolute_error(test_original[:, 2], pred_original[:, 2])
r2_yield_stress = r2_score(test_original[:, 2], pred_original[:, 2])
sqrt_mse_yield_stress = np.sqrt(mse_yield_stress)

print("Sw %:")
print(f"  Error Cuadrático Medio MSE: {mse_sw}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_sw}")
print(f"  Error Absoluto Medio MAE: {mae_sw}")
print(f"  Coeficiente de Determinación R²: {r2_sw}")

print("\nViscosidad (cP):")
print(f"  Error Cuadrático Medio MSE: {mse_viscosidad}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_viscosidad}")
print(f"  Error Absoluto Medio MAE: {mae_viscosidad}")
print(f"  Coeficiente de Determinación R²: {r2_viscosidad}")

print("\nYield stress (dinas/cm2):")
print(f"  Error Cuadrático Medio MSE: {mse_yield_stress}")
print(f"  Raíz del Error Cuadrático Medio RMSE: {sqrt_mse_yield_stress}")
print(f"  Error Absoluto Medio MAE: {mae_yield_stress}")
print(f"  Coeficiente de Determinación R²: {r2_yield_stress}")

# Calcular la importancia de las características por objetivo
for i, objetivo in enumerate(objetivos):
    print(f"Importancia de los predictores para {objetivo}:")
```

```
    # Preparar los datos de entrenamiento específicos para el objetivo actual
    y_train_objetivo = y_train.iloc[:, i

    # Entrenar el modelo con los datos específicos del objetivo
    mejor_modeloRF.fit(X_train, y_train_objetivo)

    # Obtener la importancia de las características
    feature_importances = mejor_modeloRF.feature_importances_
    indices = np.argsort(feature_importances)[::-1]

    # Mostrar las características más importantes para el objetivo actual
    for f in range(X_train.shape[1]):
        print(f"{f + 1}. {X_train.columns[indices[f]]}:
              {feature_importances[indices[f]]}")

    print()


# Gráficos de la importancia de las características
colores = ['blue', 'green', 'red']
plt.figure(figsize=(8, 10))

# Calcular la importancia de las características por objetivo
for i, objetivo in enumerate(objetivos):
    # Preparar los datos de entrenamiento específicos para el objetivo actual
    y_train_objetivo = y_train.iloc[:, i

    # Entrenar el modelo con los datos específicos del objetivo
    mejor_modeloRF.fit(X_train, y_train_objetivo)

    # Obtener la importancia de las características
    feature_importances = mejor_modeloRF.feature_importances_
    indices = np.argsort(feature_importances)

    # Mostrar las características más importantes para el objetivo actual
    plt.subplot(len(objetivos), 1, i+1)
    plt.barh(range(X_train.shape[1]), feature_importances[indices],
             align='center', color=colores[i])
    plt.yticks(range(X_train.shape[1]), X_train.columns[indices])
    plt.xlabel('Importancia')
    plt.title(f'Importancia de características para {objetivo}')
    plt.gcf().patch.set_alpha(0.0)
    plt.gca().patch.set_alpha(0.0)
    plt.tight_layout()

plt.show()
```

```
# Visualizar las predicciones con gráficos de dispersión
import matplotlib.pyplot as plt
colores = ['blue', 'green', 'red']
plt.figure(figsize=(25, 10))

for i, objetivo in enumerate(objetivos):
    plt.subplot(1, 3, i + 1)
    plt.scatter(test_original[:, i], pred_original[:, i], alpha=0.3,
color=colores[i])
    plt.plot([test_original[:, i].min(), test_original[:, i].max()],
             [test_original[:, i].min(), test_original[:, i].max()], 'k--',
lw=2)
    plt.title(f'Actual vs Predicho - {objetivo}', fontsize = 25)
    plt.xlabel('Actual', fontsize = 20)
    plt.ylabel('Predicho', fontsize = 20)
    plt.gcf().patch.set_alpha(0.0)
    plt.gca().patch.set_alpha(0.0)
    plt.grid(True)

plt.tight_layout()
plt.show()
```

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA
Departamento de Ingeniería Metalúrgica
Hoja Resumen Memoria de Título

Título: Predicción del comportamiento del espesador de relaves HRT 4001 en función de los parámetros de la pulpa de alimentación y reología en Compañía Minera Doña Inés de Collahuasi.

Nombre Memorista: Nathalia Andrea Olivera Bascur

Modalidad	Proyecto	**Profesor(es) Patrocinante(s)**
Concepto		
Calificación		
Fecha	22.10.2024	**Prof. Leopoldo Gutiérrez B.**
Prof.ª Eugenia Araneda H.		**Ingeniero Supervisor: Jorge Cortínez C.**
		Institución: Compañía Minera Doña Inés de Collahuasi

Comisión (Nombre y Firma)	
Prof. Fernando Betancourt C.	Prof. Luver Echeverry V.

Resumen

El presente proyecto de memoria de título expone el diseño de un modelo predictivo para el comportamiento del espesador de relaves HRT 4001 ubicado en la Planta Concentradora Ujina de Compañía Minera Doña Inés de Collahuasi, utilizando modelos de Machine Learning implementados con lenguaje de programación *Python*. Adicionalmente, con el modelo predictivo, se pueden identificar aquellas variables que permitan optimizar el funcionamiento del espesador y recuperación de agua.

Los tres modelos examinados corresponden a: Regresión Polinomial Multivariable, Redes Neuronales Artificiales (RNA) y Random Forest (RF). Los últimos dos modelos fueron modelados con la herramienta *GridSearchCV()* para crear modelos con el mejor desempeño posible. Las variables utilizadas para predecir el porcentaje de sólidos en la pulpa de descarga y reología (viscosidad y yield stress) incluyeron variables mineralógicas de la pulpa de alimentación (pH, nivel de arcillas y granulometría) y variables operacionales (flujo de alimentación, flujo de descarga, dosis de floculante, nivel de interfase, nivel de cama y torque).

Las métricas de rendimiento empleadas para comparar los modelos corresponden al Error Absoluto Medio (MAE), Raíz del Error Cuadrático Medio (RMSE) y el Coeficiente de determinación (R^2), que son comúnmente utilizadas en Machine Learning. Los resultados muestran que el modelo Random Forest produce mejores valores paras las tres variables objetivo, con un MAE de 0,042 para el porcentaje de sólidos en la descarga, 1,641 para la viscosidad y 7,363 para el yield stress, y un R^2 del 0,986, 0,906 y 0,981 respectivamente.

De los resultados se concluye que las características más influyentes corresponden a las arcillas y el nivel de cama. Aunque las características de la pulpa de alimentación no pueden ser modificadas por el proceso de espesamiento, el modelo predictivo permite experimentar con las demás variables, ayudando a identificar los ajustes necesarios para alcanzar los resultados objetivos, mientras se informa la reología.

Finalmente, como el modelo predictivo depende de los datos en línea, se sugiere una revisión de los sensores, principalmente del sistema de rastreo del tamaño de partículas, y la ampliación del conjunto de datos a lo largo del tiempo, entrenando el modelo con data futura, permitiendo que no pierda así su poder predictivo.