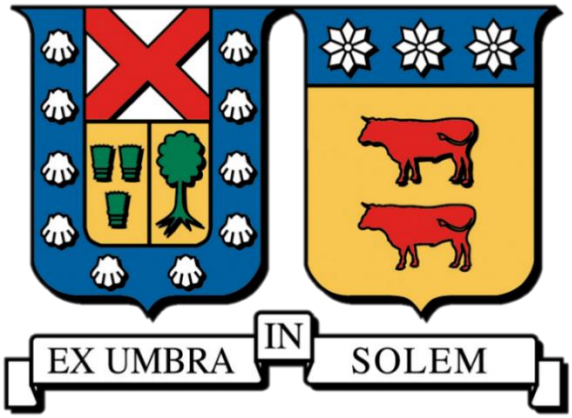
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INGENIERÍA METALÚRGICA Y DE MATERIALES
SANTIAGO – CHILE



"DESARROLLO DE UN MODELO MEDIANTE INTELIGENCIA ARTIFICIAL PARA OPTIMIZAR LA FLOTACIÓN"

ZAIRA ALEXANDRA REYES MONSALVE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL DE MINAS

Profesora guía
Francisca San Martín R.

Profesor correferente
Claudio Aguilar R.

Marzo, 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Desarrollo de un modelo mediante inteligencia artificial para optimizar la flotación

Nombre del candidato(a): Zaira Alexandra Reyes Monsalve

Carrera / Grado: Ingeniería Civil de Minas

Campus: San Joaquín **Departamento:** Ingeniería Metalúrgica y de Materiales.

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Francisca San Martin Robbiano, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☑ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

_____

_____

_____

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 10/04/2026 **Firma:** _____

Estudiante o Candidato(a):

Fecha: 7 de Abril 2026 **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

AGRADECIMIENTOS

Quiero agradecer, en primerísimo lugar, a mis padres, César Reyes y Julia Monsalve, por ser un pilar fundamental en mi vida, darme una buena educación, apoyarme en mis estudios y nunca dudar de mí. Muchas gracias por ser los mejores padres que pude pedir y apoyarme incluso en mis momentos más vulnerables. Les debo mucho en esta vida.

Les doy las gracias a las personas especiales que han estado ahí para mí. A mi primo Wilson, por ser prácticamente un hermano de distinta madre y un amigo con el que puedo contar. A los amigos que he conocido durante este camino universitario: Francisca, Gonzalo, Jamiro, Casich, Jaime, Lucas, Andrés, Allan, Javiera, Matías, Koke y Guille. A los buenos amigos externos a la universidad que han estado ahí cuando los he necesitado: Juan, Ari, Aru, Diana, a todo mi Team Queen, al Team Troll y al staff del servidor de modero, especialmente a los administradores. Gracias, amigos. Los quiero mucho a todos.

Aprovecho también este pequeño espacio para darle las gracias a mi antigua pareja. A pesar de todo, debo reconocer que él me apoyó en momentos críticos y, sin lugar a duda, me ayudó en la construcción de este trabajo de memoria. Le doy gracias por ayudarme a buscar información, por acompañarme en llamadas mientras avanzaba con esto y por nunca dudar en que yo pudiera terminar con éxito este trabajo. Esto no hubiera sido posible sin él. Le deseo lo mejor en la vida.

También a aquellos académicos y funcionarios de la universidad que apoyaron a mi crecimiento como estudiante. En especial, doy las gracias a la profesora Francisca San Martín, por entregarme este tema de memoria, por su paciencia, apoyo y guía durante este proceso de titulación. También quiero agradecer a María Berríos, la psicóloga de la universidad que me atendió durante mis últimos meses como estudiante, por haberme aconsejado, enseñado y animado en esos momentos difíciles que tuve que vivir. Aún tengo mucho que trabajar en mí misma para ser mejor y lograr estar feliz como solía ser, pero voy por buen camino gracias a mi terapia.

Finalmente, a esas personas que no nombré en esta sección, pero que, de alguna forma, fueron directa o indirectamente parte de esta etapa de mi vida. Muchas gracias a todos.

DEDICATORIA

A mis papás, por siempre apoyarme en mis mejores y peores momentos.

Este triunfo es para ustedes.

RESUMEN

La flotación de minerales es un proceso fundamental en la industria minera que permite separar minerales valiosos de otros no deseados. Este proceso es crucial para la obtención de concentrados de minerales. Por su parte, la inteligencia artificial (IA) se ha utilizado como una herramienta en los últimos años para mejorar la eficiencia de los procesos. Este tema de memoria se centró en explorar la aplicación de técnicas de IA en la optimización de la flotación de minerales.

El estudio comenzó con una introducción a la flotación de minerales y su importancia dentro de los procesos de la industria minera. Luego, se presentó una visión general de cómo se utiliza la IA en el mundo actual. Se analizaron técnicas de modelamiento con IA que se pueden aplicar para predecir la recuperación de cobre y, de esta manera, optimizar la flotación.

Se describieron los pasos para desarrollar un código en Python para predecir recuperación de cobre, desde la selección de algoritmos hasta el tratamiento de cada parámetro de flotación, donde se usaron casos reales de flotación para alimentar una base de datos capaz de brindar la suficiente información inicial para obtener predicciones.

Se lograron obtener predicciones de cobre, algunas buenas y otras deficientes. El análisis demostró que el modelo es capaz de obtener buenos resultados siempre y cuando los parámetros que se deseen ingresar tengan similitud con la database, es decir, dependerá de qué tantas veces los parámetros estén presentes en la base de datos y de la similitud con alguno de los testeos de entrada. Si se intenta predecir una recuperación con parámetros que no se encuentran presentes o son escasos en la base de datos, los resultados de recuperación serán alejados de la realidad.

Como conclusión, el modelo es capaz de entregar predicciones de recuperación de cobre, lo cual puede ayudar a obtener estimaciones aceptables de resultados de laboratorio. Sin embargo, el código es extremadamente dependiente de los datos de entrada, por lo que se recomienda contar con una base de datos robusta y que no varíe demasiado en sus parámetros.

GLOSARIO

- **A65:** Un éter de poliglicol con un peso molecular de 250 g/mol.
- **ADD:** Dibutil ditiofosfato de amonio.
- **Aero 3894:** Colector basado en tionocarbamato.
- **Aerofloat:** Reactivo químico de la familia de los ditiofosfatos en el proceso de flotación minera, utilizado como colector.
- **Aerophine 3418A:** Colector basado en ditiofosfinato de sodio.
- **B201:** Un tipo de colector desarrollado por el Instituto General de Investigación de Minería y Metalurgia de Beijing.
- **BX:** Butil xantato.
- **C7240:** Alquil ditiofosfato de sodio.
- **Data frame:** Estructura tabular con etiquetas para almacenar datos, similar a una hoja de cálculo de Excel.
- **DBS:** Ditiofosfato de butil sodio.
- **EX:** Xantato de etilo.
- **Float:** En lenguaje Python, valor numérico perteneciente al conjunto de los números reales, es decir, que posee decimales.
- **IA:** Inteligencia artificial.
- **KAX:** Xantato de amilo de potasio.
- **MIBC:** Metil isobutil carbinol.
- **MX7017:** Colector basado en tionocarbamato modificado.
- **NaIX:** Xantato isopropílico de sodio.
- **PAX:** Amil xantato de potasio.
- **PBX:** Butil xantato de potasio.
- **PX:** Xantato de pentilo.
- **R-66:** Reactivo que, según estudios, puede sustituir a la cal en la flotación.
- **S-7261A:** Polímero macromolecular desarrollado por Solvay Company, utilizado como depresante.
- **SBX:** Butil xantato de sodio.
- **SEX:** Xantato de etilo de sodio.

- **SIAX:** Xantato de isoamilo de sodio.
- **SIBX:** Isobutil xantato de sodio.
- **SIPX:** Isopropil xantato de sodio.
- **ST:** Tritiocarbonato de sodio.
- **TC o TC-1000:** Colectores basados en tionocarbamato.
- **Z11:** Xantato isopropílico de sodio.
- **α-Fe_2O:** Forma alfa del óxido de hierro III, también llamado hematita.
- **γ-Al_2O_3:** Alúmina gamma.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. General

La industria minera desempeña un papel fundamental en la economía global, proporcionando los metales y minerales necesarios para la producción de una amplia variedad de bienes y servicios. Sin embargo, enfrenta constantes desafíos en un mundo que exige mayor eficiencia, sostenibilidad y responsabilidad con el medio ambiente. La flotación se encuentra dentro de estos desafíos, siendo uno de los procesos esenciales para la separación de minerales valiosos de otros materiales no deseados.

En este contexto, la inteligencia artificial (IA) emerge como una herramienta con el potencial de revolucionar la forma en que se abordan estos desafíos en la industria minera. La IA posee una serie de funciones y herramientas diferentes que apoyan la realización de distintas funciones de diferentes áreas, cubriendo diversas necesidades del mundo moderno. Una de estas herramientas de la IA ofrece la capacidad de modelar y optimizar procesos mediante la programación y el Machine Learning, siendo este último el que permite entrenar los modelos y mejorar la efectividad de los resultados.

Este trabajo de memoria tiene como finalidad crear una base de datos que logre optimizar la flotación, utilizando datos reales recopilados para crear un modelo predictivo y empleando IA para entrenar dicho modelo. Se planea abordar desde la revisión de la bibliografía y la identificación de métodos de modelado hasta la recopilación de datos y el desarrollo de modelos predictivos, los cuales serán probados y entrenados.

La relevancia de esta investigación radica en su capacidad de mejorar la eficiencia de uno de los procesos más importantes de la industria minera. Hay que considerar que la flotación permite recuperar minerales valiosos a partir de menas de baja ley, lo que sería económicamente inviable de otra manera. Además, contribuye a la reducción de residuos, impulsando la sostenibilidad ambiental. Debido a esto, optimizar este proceso, con la finalidad de volverlo más eficiente, traería consigo impactos positivos en el procesamiento de minerales y la minería sostenible.

1.2. Objetivos del tema

A continuación, se presentan los objetivos de este tema de memoria.

1.2.1. Objetivo general

Desarrollar un modelo en el software Python con el fin de optimizar el proceso de flotación de minerales mediante inteligencia artificial y Machine Learning.

1.2.2. Objetivos específicos

- Interpretar distintos parámetros y valores de flotación de diversas fuentes literarias para evaluar tendencias en los datos.
- Ajustar parte de los datos a diversos modelos, eligiendo el mejor ajuste de los datos, dependiendo del caso o parámetro.
- Probar el código final, utilizando parte de los datos y creando testeos con parámetros aleatorios, para comprobar su efectividad.

1.3. Hipótesis

La hipótesis para este trabajo de memoria propone que los parámetros de la flotación de minerales poseen tendencias que son posibles de modelar mediante inteligencia artificial, lo que se traduce en lograr crear un apoyo para mejorar y optimizar la eficiencia del proceso.

1.4. Alcances

La eficiencia del proceso de flotación de minerales depende de muchos factores que pueden ser controlados. Para efectos de este trabajo de memoria, se evaluarán solamente parámetros relacionados a la recuperación, pH, reactivos, agua de la solución, mineral utilizado, tamaño de partícula y burbuja, tipo de celda de flotación, velocidad y tipo de gas, tiempo de flotación y tiempo de acondicionamiento. Otros parámetros que no se incluyen en este listado no serán utilizados para crear los modelos.

El desarrollo de los modelos predictivos será limitado por las herramientas que ofrece el software Python.

2. ANTECEDENTES DEL TEMA

Para este tema de memoria es importante comprender la flotación de minerales, así como los modelos predictivos posibles de crear.

2.1. Flotación de minerales

2.1.1. Características generales

La flotación de minerales es un proceso físico-químico que se utiliza como método principal de concentración en el procesamiento de minerales. Este proceso se basa en la interacción entre burbujas y partículas finas de sólido suspendidas en una solución acuosa. Las partículas de mineral son introducidas al sistema y, debido a la transferencia de energía, colisionan con las burbujas de aire presentes en la pulpa, las cuales son formadas con la ayuda de reactivos puestos en la solución. De manera selectiva, las partículas de los minerales valiosos se adhieren a las burbujas, generando un agregado conocido como "colección", la cual comienzan a flotar hacia la superficie de una celda de flotación. Los minerales de desecho, que no se adhieren a las burbujas, se hunden y se descartan. Véase Figura 1.

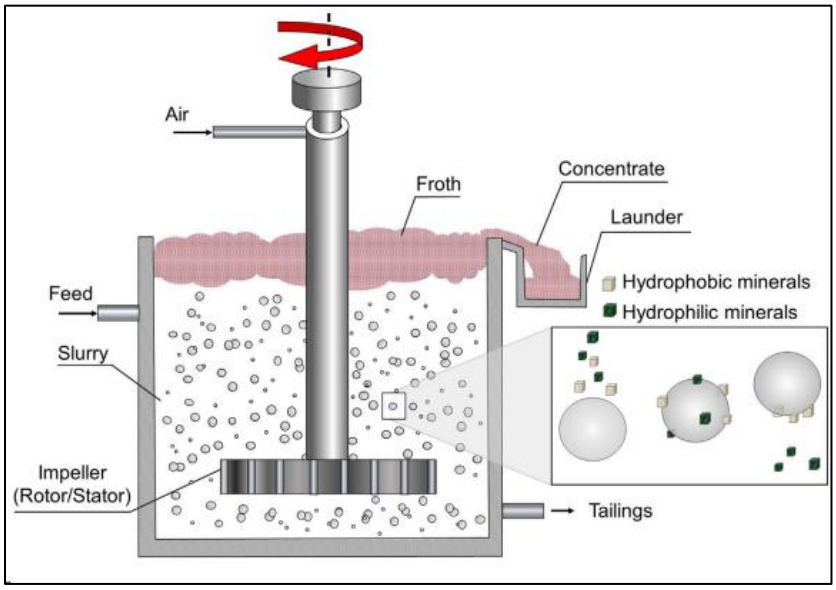


Figura 1: Ilustración del proceso de recuperación de mineral valioso mediante flotación (Wills. & Finch, 2016).

2.1.2. Tipos de flotación

Se puede observar cómo se comporta la recuperación de mineral en función de la ley del proceso. Véase Figura 2. Esto significa que, si se decide tener alta recuperación del mineral valioso, se recuperará también ganga, lo cual bajará la ley del concentrado. Por otra parte, si se decide priorizar la ley del concentrado, evitando que se recupere ganga, esto dará paso a no poder recuperar todo lo valioso.

Esto es posible solucionarlo creando circuitos de flotación, los cuales constarán de, principalmente, 3 etapas.

- Flotación rougher: La flotación rougher es la primera etapa del proceso y se enfoca en la recuperación inicial de minerales valiosos, pero a menudo es seguida por etapas de limpieza y scavenger para mejorar la calidad del concentrado final. Esta etapa es importante para lograr una buena operación minera, ya que determina en gran medida el rendimiento y la eficiencia de la concentración de minerales.
- Flotación cleaner: Su objetivo principal es mejorar la calidad y pureza del concentrado obtenido en la etapa rougher. En esta fase, se realizan una serie de etapas adicionales de flotación con reactivos selectivos, con el fin de eliminar impurezas o minerales no deseados que puedan estar presentes en el concentrado inicial. De esta manera, la flotación cleaner contribuye a obtener un producto final de mejor calidad y mayor concentración de los minerales valiosos.
- Flotación scavenger: La flotación scavenger es una etapa posterior en el proceso de concentración de minerales, que sigue a la flotación rougher y cleaner. Su función principal es recuperar cualquier mineral valioso que pueda haber quedado atrapado o no recuperado en las etapas anteriores del proceso. En esta fase, se emplean reactivos especiales y se lleva a cabo una flotación adicional para maximizar la recuperación de minerales valiosos, lo que contribuye a reducir las pérdidas y garantizar una mayor eficiencia en la operación de concentración de minerales. La flotación scavenger es esencial para aprovechar al máximo el valor de los minerales y optimizar el rendimiento de la planta de procesamiento.

Se puede apreciar un circuito RCS de flotación con estas 3 etapas. Véase Figura 3.

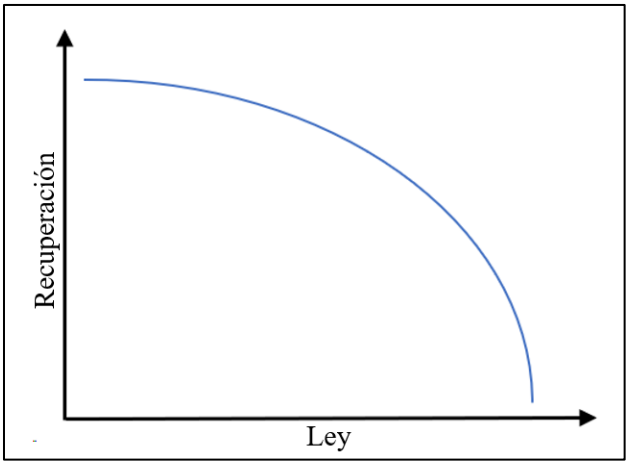


Figura 2: Gráfico Recuperación v/s Ley en el proceso de flotación.

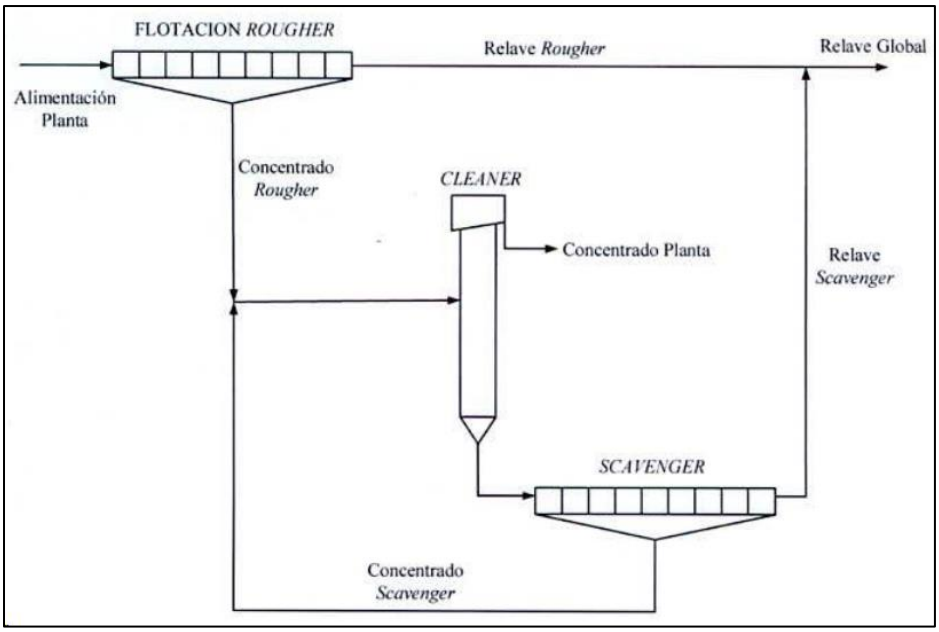


Figura 3: Circuito RCS (Yianatos, & Vinnett, 2015).

2.1.3. Parámetros generales de la flotación

A continuación, se enlistan y definen algunos parámetros que son importantes considerar si se quiere optimizar la eficiencia de la flotación.

- **pH**

El pH es un parámetro crítico en el proceso de flotación de minerales, y su control es esencial para lograr una separación efectiva y selectiva de los minerales. La alcalinidad de la pulpa, es decir, el nivel de acidez o basicidad desempeña un papel fundamental en la flotación. Esto se debe a que el pH influye en la carga superficial de las partículas minerales, lo que a su vez regula las interacciones con los reactivos de flotación, como los colectores iónicos y los agentes reguladores. En la práctica, lograr la selectividad en separaciones complejas implica encontrar un equilibrio delicado entre las concentraciones de estos reactivos y el pH de la pulpa (Wills & Finch, 2016).

- **Reactivos**

Los reactivos, también denominados reguladores o modificadores, se convierten en herramientas estratégicas en la flotación, permitiendo a los operadores ajustar la selectividad de la adhesión del colector a las partículas minerales. Este ajuste selectivo es vital, ya que puede conducir a la recuperación óptima de minerales valiosos, al tiempo que minimiza la adhesión de minerales no deseados (Wills & Finch, 2016).

Los reactivos pueden clasificarse en diversas categorías según su función específica:

- Colector: Los colectores son agentes químicos diseñados para conferir a ciertos minerales la capacidad de repeler el agua, lo que es esencial para que estos minerales puedan separarse de las impurezas y otros minerales no deseados durante la flotación. Es importante comprender que la hidrofobicidad, es decir, la aversión al agua, es un requisito crucial en la flotación, ya que permite que los minerales de interés se adhieran a las burbujas de aire y asciendan a la superficie, donde pueden ser recolectados y separados del resto de la pulpa (Wills & Finch, 2016).
- Depresante: Los depresantes en el contexto de la flotación minera mejoran la selectividad del proceso. Su función principal es convertir ciertos minerales en hidrófilos, lo que a su vez evita que se adhieran a las burbujas de aire y floten. Esta estrategia es esencial para lograr una flotación económica y efectiva de minerales específicos en una mezcla. Existen varios mecanismos a través de los cuales los

depresantes logran este efecto deseado, y en muchos casos, se combinan varios de ellos. Estos mecanismos incluyen: adsorción de especies hidrófilas, bloqueo de sitios de adsorción del colector, eliminación (desorción) de especies activadoras (desactivación) y eliminación de sitios hidrófobos (desorción/destrucción del colector adsorbido) (Wills & Finch, 2016).

- Espumante: Wills y Finch (2016) definen los espumantes como componentes clave en el proceso de flotación minera, desempeñando un papel crucial en la formación y estabilización de la espuma utilizada para separar minerales valiosos de minerales no deseados. Dentro de sus funciones a destaca:
    - Ayudar a la formación y conservación de burbujas pequeñas: Los espumantes ayudan a crear burbujas de aire de tamaño adecuado. Estas burbujas son esenciales para la flotación, ya que aumentan la probabilidad de colisión entre las burbujas y las partículas minerales.
    - Reducen la velocidad de ascenso de las burbujas: Controlar la velocidad de ascenso de las burbujas es vital para asegurar que pasen suficiente tiempo en la pulpa para interactuar con las partículas. Los espumantes desaceleran el ascenso de las burbujas, lo que prolonga su tiempo de residencia en la pulpa y aumenta las posibilidades de colisión con las partículas minerales.
    - Ayudar a la formación de espuma: La formación de una espuma estable es esencial para recolectar las partículas minerales flotantes en la superficie de la pulpa. Los espumantes evitan que las burbujas revienten cuando llegan a la superficie, lo que garantiza que las partículas atrapadas en la espuma se separen del resto de la pulpa y se puedan recoger como producto de flotación.

Otros tipos de reactivos pueden ser:

- Activadores: Son sustancias químicas diseñadas para modificar la superficie de los minerales, permitiendo así que interactúen de manera efectiva con los colectores. Su función principal radica en transformar la naturaleza química de las partículas minerales, haciéndolas hidrófobas, es decir, repelentes al agua. Este cambio en la afinidad de la superficie mineral es fundamental para la separación selectiva de

minerales valiosos de los residuos no deseados, un paso crítico en la industria minera (Wills & Finch, 2016).

- Modificadores de pH: Estos reactivos sirven para poder estabilizar la acidez de la pulpa en un pH determinado, brindando así el ambiente adecuado para que el proceso de flotación se efectúe con eficiencia (CODELCO, 2019).

- **Propiedades superficiales del mineral**

Yianatos y Vinnett (2015) mencionan que la flotación en la minería es un proceso que se basa en la explotación de las diferencias en las propiedades superficiales de los minerales para separar y concentrar el mineral valioso de otros componentes no deseados. Estas propiedades superficiales son esenciales para el éxito del proceso y dependen de varios factores clave:

a) Naturaleza del mineral: Cada tipo de mineral presenta características superficiales únicas que influyen en su capacidad para interactuar con los reactivos de flotación y las burbujas de aire. Estas diferencias pueden estar relacionadas con la composición química, la estructura cristalina y otras propiedades inherentes.

b) Heterogeneidad: La distribución y disposición de los minerales en la mena son cruciales. La liberación, diseminación y asociación de los minerales en la matriz de la mena afectan la eficiencia de la flotación. Los minerales que están bien liberados de la ganga y que se asocian de manera efectiva con las burbujas de aire tienen más posibilidades de ser recuperados.

c) Forma de las partículas: La geometría de las partículas minerales desempeña un papel importante. Partículas de distintas formas pueden interactuar de manera diferente con las burbujas de aire y los reactivos de flotación. Por ejemplo, partículas esféricas pueden tener un comportamiento distinto en comparación con partículas más angulares.

d) Topografía de las partículas (rugosidad): La rugosidad de la superficie de las partículas influye en su capacidad para adherirse a las burbujas de aire. Partículas con

superficies más rugosas pueden proporcionar sitios de anclaje adicionales para las burbujas, lo que aumenta la probabilidad de flotación.

- **Tipo de celda**

Una celda de flotación es un equipo utilizado para separar minerales valiosos de minerales no deseados a través de un proceso de flotación.

Existen varios tipos de celdas de flotación, cada una con sus propias características y aplicaciones. Algunos de los tipos más comunes de celdas de flotación incluyen:

- Celdas de flotación mecánicas

Estas celdas utilizan agitación mecánica mediante un sistema de paletas o impulsores para mezclar la pulpa y el aire. Las celdas mecánicas se utilizan comúnmente en operaciones de flotación a gran escala y son conocidas por su robustez y capacidad para tratar grandes volúmenes de pulpa.

- Celdas de flotación neumáticas

En este tipo de celdas, se utiliza aire comprimido para dispersar finas burbujas de aire en la pulpa. Estas burbujas se adhieren a las partículas de mineral, permitiendo la flotación. Las celdas de flotación neumáticas son eficientes en la flotación de partículas finas y son ampliamente utilizadas en aplicaciones de flotación de minerales de alto valor.

- Celdas de flotación por columna

Las celdas de flotación por columna son equipos verticales que utilizan una corriente ascendente de agua y aire para separar minerales. Son especialmente eficaces en la flotación de partículas finas y se utilizan a menudo en aplicaciones donde la selectividad y la limpieza del concentrado son fundamentales.

- **Tamaño de partícula**

Yianatos y Vinnett (2015) determinan que la definición del tamaño de partículas es fundamental para comprender y optimizar el proceso de recuperación de minerales. El

tamaño de las partículas en la corriente de alimentación de los equipos de flotación desempeña un papel crítico en la eficiencia de la operación.

En este escenario, se observa que la recuperación de mineral alcanza su máximo en un rango de tamaño típicamente entre 50-100 micrómetros, disminuyendo para partículas más pequeñas y más grandes de este rango. Las implicaciones de distintos tamaños de partículas son los siguientes:

- Partículas Finas

Las partículas finas tienden a disminuir la recuperación en la flotación debido a factores hidrodinámicos. Estas partículas pequeñas tienen baja inercia y, por lo tanto, se mueven con mayor facilidad junto con el fluido en el proceso. Para mejorar la cinética de flotación de partículas finas, se pueden utilizar burbujas más finas. Sin embargo, la reducción del tamaño de las burbujas puede no ser adecuada cuando se trata de minerales con una distribución heterogénea de tamaños de partículas, ya que podría perjudicar la recuperación de partículas gruesas. Además, existen otros desafíos, como el arrastre de burbujas pequeñas hacia las colas y la pérdida de la interfaz entre la pulpa y la espuma. Otra estrategia para mejorar la recuperación de partículas finas implica promover la coagulación o floculación selectiva del mineral, como sugiere Sivamohan (1990).

- Partículas gruesas

Las partículas gruesas presentan desafíos distintos en la flotación. Tienen un menor grado de liberación, menos tiempo de residencia en los equipos de flotación y una eficiencia de recolección más baja. Uno de los principales problemas con las partículas gruesas es su tendencia a romper el agregado partícula-burbuja debido a la turbulencia presente en el interior de los equipos de flotación. Las partículas en la superficie de las burbujas están sujetas a fuerzas de cizalladura y arrastre, y se desprenden si estas fuerzas superan la tensión superficial que mantiene unida la partícula a la burbuja. Además, debido a su menor fuerza de adhesión, las partículas gruesas tienen más probabilidades de desprenderse en la zona de espuma. Este desprendimiento puede deberse a la coalescencia

o el colapso de las burbujas, lo que implica que las partículas retornen a la zona de recolección.

Se muestra cómo se comporta la recuperación de ciertos minerales en base al tamaño de partícula. Véase Figura 4.

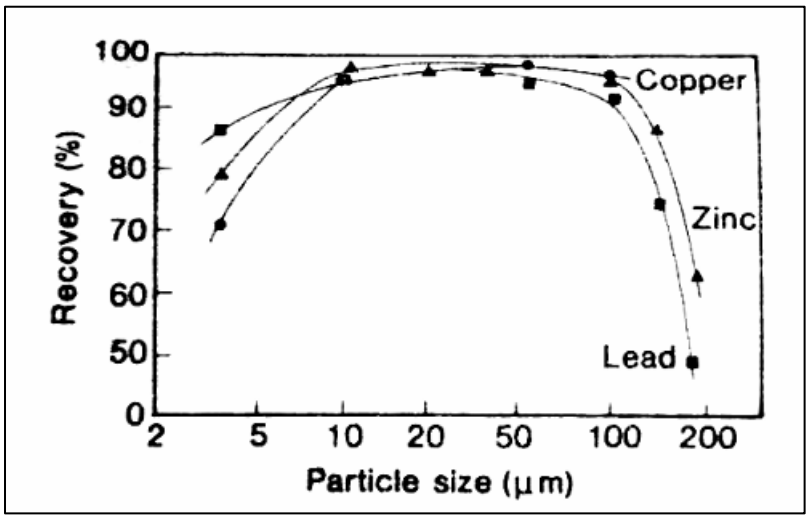


Figura 4: Variación de la recuperación de flotación con el tamaño de partícula en concentradores industriales (King, 1982).

Para mejorar la recuperación de partículas gruesas, es esencial contar con un sistema que mantenga las partículas en suspensión y disperse el gas sin generar turbulencia excesiva. Una alternativa prometedora consiste en utilizar accesorios independientes para la dispersión de la pulpa y la generación de burbujas.

Para lograr esta separación, es necesario fragmentar y reducir el tamaño de las rocas extraídas de la mina a través de las etapas de chancado y molienda. Estas etapas son fundamentales en el proceso de concentración de minerales, ya que la falta de liberación adecuada disminuye la recuperación de minerales valiosos. En otras palabras, si los minerales valiosos no se liberan de la ganga, no se podrán recuperar de manera efectiva. Véase Figura 5.

La liberación se logra al reducir el tamaño de las partículas a un nivel inferior al del grano de las especies minerales. Esto implica descomponer las rocas en fragmentos más pequeños para exponer la mayor cantidad de mineral valioso posible. Para aumentar la hidrofobicidad de

los minerales flotables (es decir, su capacidad de repeler el agua), se acondicionan con la adsorción de reactivos químicos, mientras que la ganga se mantiene preferentemente hidrófila (atraída por el agua). Este proceso asegura que la superficie de los minerales valiosos quede expuesta y sea receptiva a la adsorción de los reactivos para su posterior colección.

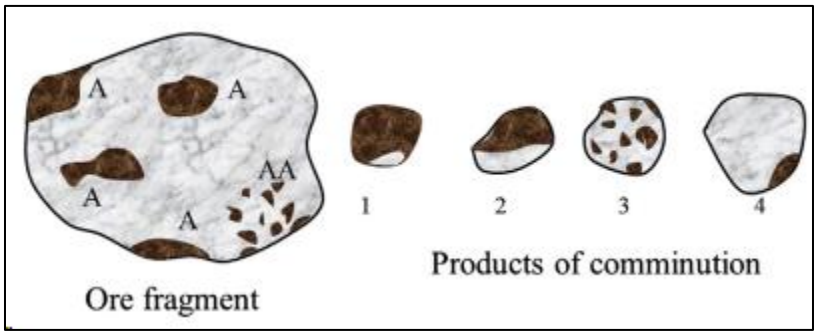


Figura 5: Producto de la conminución (Wills & Finch, 2016).

- **Tipo de agua**

El agua es un elemento importante en la flotación. Liu et al. (2013) menciona que la flotación es más efectiva en agua limpia. Sin embargo, a medida que los recursos hídricos comienzan a escasear y se exige cada vez más reducir la extracción de agua dulce, la minería ha incrementado la reutilización del agua y el acceso a muchas otras fuentes de agua para el procesamiento de minerales, con el fin de ahorrar agua dulce, especialmente en la flotación por espuma, ya que en algunas ocasiones el agua salada puede actuar como un espumante. La implementación de estas estrategias podría afectar en la calidad del agua y, a su vez, la eficiencia de la flotación.

La salinidad del agua es importante, pues si bien puede ayudar a obtener buena recuperación de ciertos minerales, la presencia de iones puede generar precipitaciones que afectan a ciertos reactivos.

Es por esto que, actualmente, es importante el tratamiento del agua en la minería, pues su salinidad y iones puede afectar negativamente a la recuperación.

- **Tamaño de burbuja**

El tamaño de las burbujas influye en la posibilidad de captura de partículas de mineral, lo que afecta la eficiencia de la flotación. El espumante, colector, las propiedades de la pulpa, la energía mecánica aplicada y el caudal de aire son los principales parámetros que influyen en el tamaño de las burbujas (Wang et al, 2020).

Este parámetro es dependiente del tamaño de partículas. Al aumentar el tamaño de partícula, existe una disminución en la probabilidad de adhesión y estabilidad en el conjunto burbuja-partícula, impactando de forma negativa la recuperación de partículas gruesas. Esto se puede equilibrar si se aumenta el diámetro de burbuja, lo cual aumenta nuevamente la probabilidad de adhesión entre estos dos elementos.

- **Velocidad del gas**

La velocidad superficial del gas se mide con la siguiente ecuación:

$$J_g = \frac{Q_g}{A}$$

Donde:

- J_g: Velocidad superficial (cm/s).
- Q_g: Flujo de gas (cm^3/s).
- A: Área transversal de la celda.

Según Metso, la cantidad de flujo de gas afecta directamente el rendimiento y, por ende, la recuperación de la flotación. El aumento de la alimentación de gas produce más burbujas, lo que conduce a una mayor probabilidad de interacción burbuja-partícula. Sin embargo, el exceso de este flujo de gas puede reducir la calidad del concentrado y también provocar que se extraiga más agua de la celda en la capa de espuma debido al aumento más rápido de las burbujas.

- **Tipo de gas**

El tipo de gas en la flotación afecta la eficiencia de separación al controlar el potencial redox y la hidrofobicidad superficial (Shen, 2022). Además, influye en el Potencial Zeta de las burbujas, la tasa de difusión y la formación de iones que pueden provocar precipitaciones, siendo el oxígeno necesario para sulfuros, mientras que el nitrógeno es un gas inerte adecuado.

El gas elegido puede ayudar a oxidar o, por el contrario, detener la oxidación. También puede aportar a controlar el pH, ser capaces de crear más burbujas, influir en la densidad de las burbujas y, a su vez, en el ascenso de estas.

El aire es uno de los gases más utilizados en este proceso, el cual posee principalmente oxígeno (O_2) y nitrógeno (N_2).

- **Tiempo de flotación**

El tiempo de flotación se refiere al período durante el cual se someten las partículas de mineral a un proceso de flotación en una celda de flotación para lograr la separación de los minerales valiosos de los minerales no deseados. Este tiempo es un parámetro importante en el diseño y operación de las plantas de flotación, ya que afecta directamente la eficiencia y el rendimiento del proceso.

- **Tiempo de acondicionamiento**

El tiempo de acondicionamiento es el período durante el cual las pulpas de mineral y reactivos químicos se mezclan y se mantienen en una celda o tanque de acondicionamiento antes de introducirlas en las celdas de flotación. Este paso es crucial en el proceso de flotación y tiene un impacto significativo en la eficiencia y selectividad de la separación de minerales.

2.2. Inteligencia Artificial y Machine Learning

La inteligencia artificial ha revolucionado a la sociedad en la actualidad. Sin embargo, no existe una sola definición que permite explicar lo que es realmente (Rusell & Norvig, 2004).

Mientras que una parte se centra en lo mental y razonal, la otra se enfoca en la conducta. Asimismo, un enfoque evalúa lograr la semejanza con el comportamiento humano, mientras que el otro, busca ser racional. Juntando estas ideas, la IA tiene 4 principales enfoques.

- Pensar como humanos.
- Actuar como humanos
- Pensar racionalmente.
- Actuar racionalmente.

Dependerá de las necesidades de cada individuo qué conseguir con la IA.

2.2.1. Aprendizaje estadístico

Según James et al. (2013), el análisis estadístico es un proceso de recolección, organización, análisis, interpretación y presentación de datos para descubrir patrones, relaciones y tendencias. Implica el uso de métodos y técnicas estadísticas para sacar conclusiones y hacer inferencias sobre una población a partir de una muestra. El análisis estadístico se utiliza en diversos campos, incluyendo las ciencias sociales, los negocios, la salud y la investigación. Ayuda a investigadores y tomadores de decisiones a tomar decisiones informadas, identificar tendencias, probar hipótesis y hacer predicciones. Algunos propósitos clave del análisis estadístico incluyen:

- Análisis descriptivo: Resumir y describir datos utilizando medidas como media, mediana y desviación estándar.
- Análisis inferencial: Hacer inferencias y sacar conclusiones sobre una población a partir de una muestra.
- Prueba de hipótesis: Evaluar la significación de las relaciones o diferencias entre variables.

- Modelado predictivo: Desarrollo de modelos para predecir resultados futuros basados en datos históricos.

El análisis estadístico proporciona un enfoque sistemático y objetivo para comprender e interpretar los datos, lo que permite a los investigadores tomar decisiones basadas en la evidencia. Se puede ver un ejemplo de este análisis, en el cual los datos obtenidos pueden ser modelados mediante una función. Véase Figura 6. Si bien los datos no coinciden exactamente con la función, se ajustan de tal forma que permiten inferir el comportamiento de los datos y predecir tendencias. Cada dato tiene un error asociado, el cual corresponde a su cercanía con la función modelada.

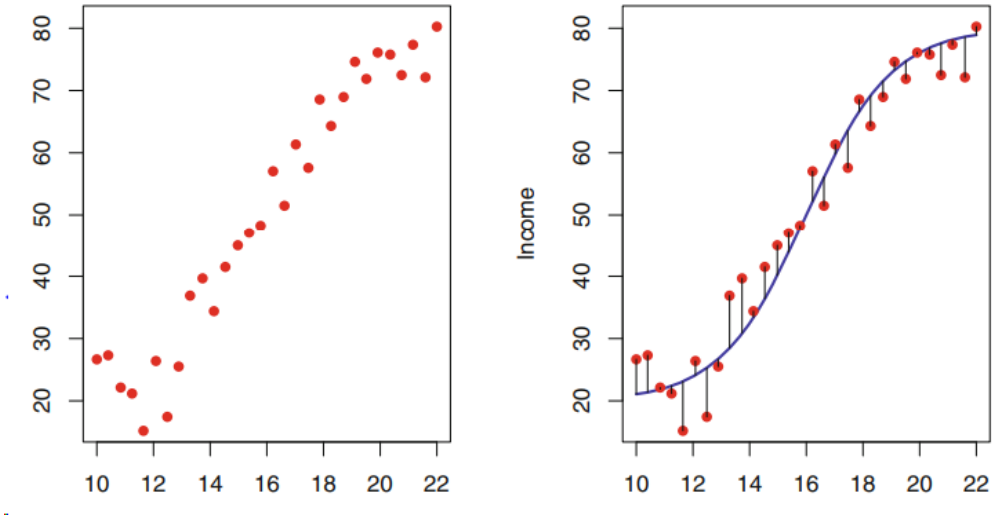


Figura 6: Set de datos a la izquierda. Set de datos con función modelada a la derecha (James, G. et al. 2013).

2.2.2. Clasificación

La clasificación en Machine Learning, como indica su nombre, consta de clasificar los datos en categorías, dependiendo de sus parámetros de entrada. Existen diversos modos de clasificación (Belcic, I.).

- **Clasificación binaria:** Clasifica los datos en dos categorías exclusivas.
- **Clasificación multiclase:** Clasifica los datos en más de dos categorías exclusivas.

- **Clasificación multietiqueta:** Clasifica los datos en categorías no exclusivas, es decir, los datos pueden pertenecer a más de una categoría.
- **Clasificación desequilibrada:** Tiene una distribución desigual de puntos de datos entre categorías.

2.2.3. Algoritmos de clasificación y predicción

- **Árbol de Decisión**

Los árboles de decisión son una técnica de Machine Learning que se utiliza para resolver problemas de clasificación y regresión. Estos árboles representan un modelo de toma de decisiones que se asemeja a una estructura de árbol con nodos y ramas. Cada nodo en el árbol representa una pregunta o una característica del conjunto de datos, y las ramas se dividen en función de las respuestas a esas preguntas. Véase Figura 7 como ejemplo.

El objetivo principal de un árbol de decisión es dividir el conjunto de datos en subconjuntos más pequeños de manera que la clasificación o predicción de una muestra sea lo más precisa posible. Esto se logra mediante la selección de características que mejor separan las clases o predicen valores en el conjunto de datos.

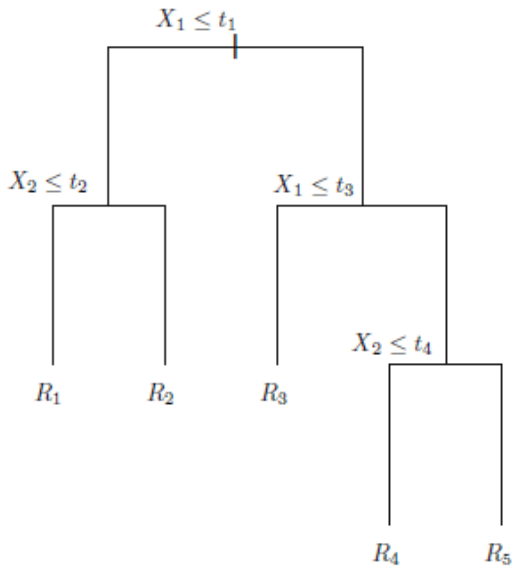


Figura 7: Caso de modelamiento por árboles de decisión (James et al. 2013).

- **Random Forest**

Random Forest es un método de conjunto (ensemble method), lo que significa que combina varios modelos para obtener una mejor predicción final. En este caso, combina múltiples árboles de decisión y promedia (o, en otros casos, vota) sus resultados para reducir errores y mejorar la precisión (Tardivon, 2022).

Un modelo de Random Forest puede estar compuesto por decenas o incluso cientos de árboles de decisión. Cada árbol de decisión se entrena con un subconjunto aleatorio de los datos y genera su propia predicción (por ejemplo, "sí" o "no"). Luego, todas esas predicciones se combinan para obtener el resultado final. En problemas de clasificación, cada árbol "vota" y la respuesta final es la que obtiene la mayoría.

Este procedimiento se conoce como bagging (bootstrap aggregating). Consiste en:

- Crear varios subconjuntos aleatorios de los datos originales.
- Entrenar un modelo independiente con cada subconjunto.
- Combinar las predicciones de todos los modelos (por ejemplo, mediante votación o promedio).

Así, como indica Tardivon (2022), aunque cada árbol individual no sea perfecto, al combinar muchos de ellos se obtiene un modelo más estable, preciso y robusto.

Él método se presenta de forma visual. Véase Figura 8.

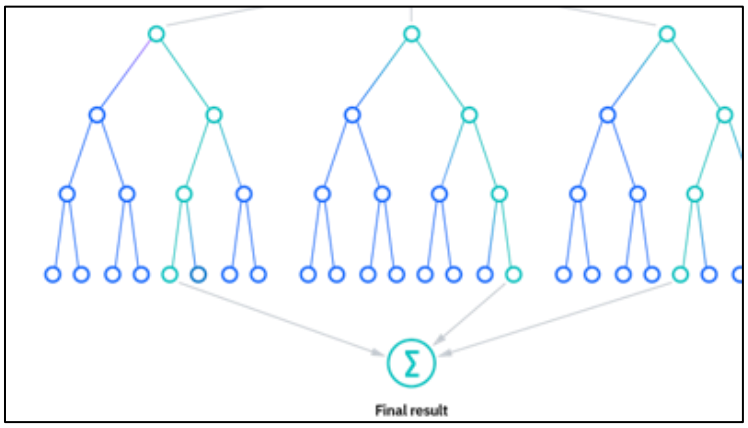


Figura 8: Cómo funciona el algoritmo de bosque aleatorio.

3. METODOLOGÍA

3.1. Recopilación de datos

Para establecer un objetivo claro, se decidió optar por una predicción de recuperación de cobre, siendo eso lo que se analizará posteriormente.

Se comenzó reuniendo información de distintas literaturas sobre los parámetros requeridos sobre flotación de minerales para poder crear la data base. Se consideraron procesos únicos de flotación y no circuitos completos, debido a la diferencia que parámetros que podían tener flotaciones de un mismo circuito. Tampoco se consideró el tipo de flotación (rougher, cleaner o scavenger). A continuación, se enlistan los valores utilizados:

- Recuperación.
- pH.
- Tipo de colector.
- Tipo de depresante.
- Tipo de espumante.
- Otros reactivos.
- Dosis de colector.
- Dosis de depresante.
- Dosis de espumante.
- Dosis de otros reactivos.
- Tipo de agua.
- Tipo de mineral.
- Tipo de celda.
- Tamaño de partícula.
- Tamaño de burbuja.
- Velocidad gas.
- Tipo de gas.
- Tiempo de flotación.
- Tiempo de acondicionamiento.

Se creará la database con estos ensayos encontrados, rellenando una tabla en Excel donde los parámetros estarán enlistados en la primera fila, mientras que el número de ensayo irá en la primera columna. Si un testeo no presenta ciertos parámetros, esos se dejarán en blanco en la respectiva celda.

Véase Anexo B para el detalle de la database.

3.2. Desarrollo de los algoritmos

Con la base de datos ya creada utilizando los tests encontrados, se dispuso a escoger los algoritmos para utilizar, dependiendo del parámetro a evaluar, la tendencia que poseían los distintos datos y las semejanzas que pudieran existir entre los diversos procesos de flotación.

Se creará un código en Python para poder predecir la recuperación de cobre, siendo esta la manera de optimizar el proceso.

Véase Anexo A para el código completo.

3.3. Verificación del modelo

Con el modelo ya creado, se intentará predecir recuperaciones para confirmar su correcto funcionamiento.

En caso de que el código sí pueda predecir, se harán dos tipos de experimentos como para comprobar su efectividad:

- Predecir la recuperación de datos que ya son parte de la database.
- Predecir ensayos nuevos, creados con parámetros aleatorios.

Con las pruebas realizadas, será posible comprobar si, efectivamente, el modelo logra predecir de forma eficiente la recuperación de cobre y, por ende, optimizar la flotación de los minerales.

4. RESULTADOS Y DISCUSIONES

4.1 Interpretación de parámetros

4.1.1. Consideraciones iniciales

Para este experimento se tomaron en total 98 testeos de diferentes documentos (véase Anexo B), de los cuales 68 son parte del mismo documento y sus datos poseen similitudes entre sí, algo importante para el entrenamiento del modelo.

En esta ocasión, se ha descartado el parámetro "Tamaño de burbuja" debido a que ningún testeo encontrado poseía ese dato.

También se descartó "Tipo de gas", puesto que el único gas utilizado en los testeos que mencionan este parámetro era aire. Al no haber ninguna diferencia de gases, este parámetro no marcaría ninguna relevancia en la base de datos.

4.1.2. Recuperación

De los 100 datos encontrados, solo 98 poseen un valor de recuperación de cobre asociado, por lo cual solo esos 98 ensayos se consideraron para este trabajo de memoria. Se muestra un análisis estadístico de las recuperaciones, donde se ve que las recuperaciones se centran en el rango de 47,02% y 95,41%. Véase Tabla 1.

La recuperación de cobre se tratará de manera porcentual y con dos decimales en la database.

Tabla 1: Análisis estadístico de todas las recuperaciones presentes en la base de datos.

Cuenta	98
Media	65,97%
Error típico	0,01247767
Mediana	63,00%
Moda	62,00%
Desviación estándar	0,12352262
Varianza de la muestra	0,01525784
Rango	0,4839
Mínimo	47,02%
Máximo	95,41%

4.1.3. pH

El parámetro de pH se tratará en el código como un float, ya que algunos de estos valores poseen decimales. Sin embargo, la mayoría son números enteros entre 7 y 12. Se cuenta con 86 testeos en total que poseen este parámetro y, a su vez, 84 que poseen los parámetros de recuperación y pH a la vez, donde se presenta una curva de promedios para cada valor de pH. Véase Figura 9.

Considerando que el pH 8 solo tiene un único valor y los pHs 7 y 9 poseen menos de 10, es importante señalar que esa sección de la curva no presenta la misma igualdad de condiciones que la segunda mitad, donde los promedios podrían ser más representativos. En esta segunda mitad, se aprecia un aumento de la recuperación general a medida que el pH es mayor.

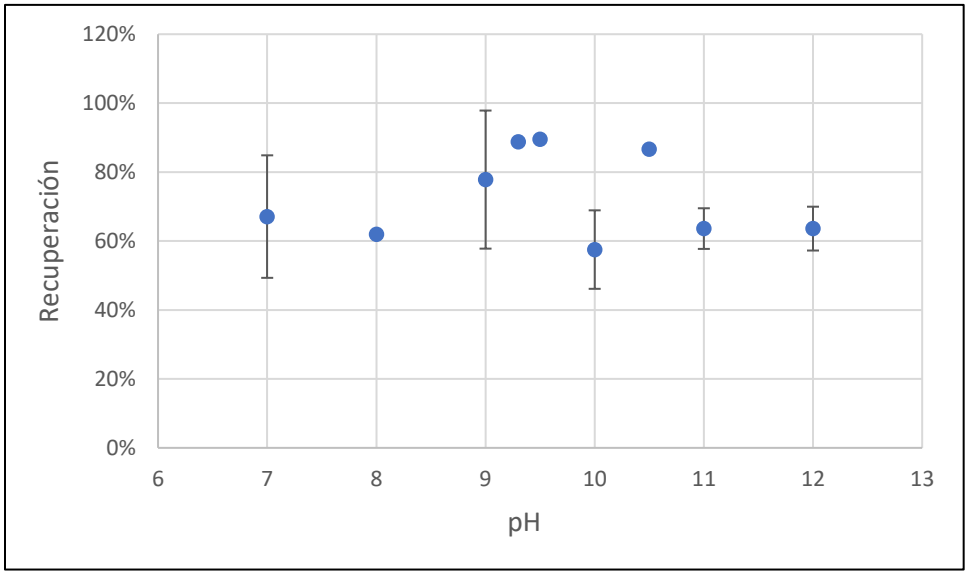


Figura 9: Recuperación vs. pH promedio de los datos, incluyendo una barra de error por desviación estándar.

4.1.4. Reactivos y dosis

Hay diversos parámetros relacionados con los reactivos. Cada grupo de reactivos, es decir, colectores, depresantes, espumantes y otros reactivos que puedan estar presente en los testeos, poseen 3 elementos para su evaluación: Tipo de reactivo, su respectiva dosis en valor numérico y su unidad de medición.

Estos elementos están correlacionados entre sí en el código: Si dos testeos poseen el mismo reactivo, automáticamente se relacionarán, especialmente si sus dosis son iguales o cercanas numéricamente; por otro lado, si dos testeos poseen distintos reactivos, no se relacionarán, aunque coincidan en la cantidad de dosis utilizada.

Los tipos de reactivos son parámetros teóricos que serán identificados de la base de datos como catálogo de entrada y, al momento de predecir con nuevos ensayos, el código mostrará una lista de los reactivos provenientes de esta database y se preguntará al usuario si el reactivo es alguno presente en la lista. En caso de que no, es posible seleccionar la opción "Otro" y agregar un nuevo reactivo. Es importante señalar que el código hace distinción de caracteres (mayúsculas, tildes, espacios, etc.), así que se deben añadir nuevos parámetros teniendo esto presente. Para futuras predicciones, los reactivos añadidos posteriores a iniciar el código pasarán a formar parte del catálogo.

La manera en que se relacionan los tipos de reactivos con sus dosis son las siguientes: Si se tiene solo un reactivo en uno de los parámetros, por ejemplo, solo un colector, debe aparecer únicamente una dosis, la cual corresponde a la dosis de ese colector. Si, por el contrario, se tienen dos o más reactivos para un parámetro, se tomarán las siguientes consideraciones:

- En la database, todos los reactivos estarán separados por comas. Por ejemplo, si existen dos colectores, la entrada seguirá un formato *X, Y*.
- Si solo existe una dosis asociada para los reactivos, significa que en esa dosis están contenidos todos los reactivos en partes iguales. Siguiendo el ejemplo del punto anterior, si se tienen dos colectores *X, Y* y una única dosis *Z*, significa que *Z* es la dosis total de X e Y juntos, siguiendo una proporción 1:1. Es importante señalar que esto solo se utilizará si el documento original lo señala de esta manera, pues estas concentraciones no siempre son aditivas. Si no es el caso, se recomienda el siguiente punto.
- Si existen las dosis asociadas para cada reactivo, deben seguir el mismo formato de entrada que los tipos de reactivos. Por ejemplo, para dos reactivos *X, Y* se deben tener dos dosis siguiendo un formato *Z, W*. Ambas dosis deben ser trabajadas con la misma

unidad, ya que eso se pregunta de forma única solo una vez, así que el usuario debe considerar cualquier transformación previa antes de rellenar la data base.

A continuación, se hablará de cada reactivo presente en el experimento.

- Colectores

De los 98 testeos, 97 presentan al menos un colector dentro de sus parámetros. A su vez, 92 tienen como colector algún tipo de xantato. También se aprecian otros colectores encontrados. Véase Tabla 2.

Tabla 2: Número de veces en que los colectores están presentes en los testeos.

Xantatos (todos los tipos)	92
DBS	68
Aerophine 3418A	3
ADD	2
TC	2
Otros	9

- Depresantes

Los depresantes son los que poseen menor presencia en los testeos encontrados, siendo solo 20 ensayos los que presentan al menos un tipo de depresante utilizado. Se pueden encontrar otros tipos de depresantes. Véase Tabla 3.

Tabla 3: Número de veces en que los depresantes están presentes en los testeos.

Na2S	7
Na2SiO3	7
Cal	5
ZnSO4	3
Otros	7

- Espumantes

Al igual que los colectores, solo 97 ensayos presentan uno o más espumantes, siendo el MIBC el más utilizado. Véase Tabla 4.

Tabla 4: Número de veces en que los espumantes están presentes en los testeos.

MIBC	78
Nasfroth	68
Aceite de terpineol	6
T-92	5
Hydrofroth	3
Otros	5

- Otros reactivos

En esta sección, hay otros tipos de reactivos que no cumplen la función de colector, espumante o depresante. Los principales son activadores o reguladores de pH, dependiendo de lo que se busque. En total, son 84 los ensayos que poseen al menos uno de estos reactivos.

- Dosis y unidades de medición

Las dosis serán valores numéricos tratados en el código como float, en caso de que puedan existir valores decimales. Además, en caso de coincidir el reactivo, mientras más cercanos sean los valores numéricos de dosis de dos o más testeos, más influirán para la predicción.

Por otro lado, las dos unidades utilizadas para estos parámetros serán g/t y kg/t.

4.1.5 Tiempo de flotación y de acondicionamiento

Los valores de tiempo se tratarán de igual manera. En el código, podrán expresarse en forma entera o decimal (float), en caso de ser necesario. Además, para simplificar los cálculos, los tiempos siempre se expresarán en minutos, así que el usuario deberá ingresar siempre los tiempos en esta unidad por defecto.

Son 93 los ensayos que poseen el parámetro de tiempo de flotación, mientras que 90 son los que poseen tiempo de acondicionamiento. Ambos tiempos se presentan en un rango entre 3 a 60 minutos.

4.1.6. Parámetros teóricos: Tipo de agua, tipo de mineral y tipo de celda

Al igual que los tipos de reactivos, estos valores serán interpretados como texto, haciendo distinción de caracteres.

- Tipo de agua

Son 90 los ensayos que presentan al menos 1 tipo de agua, siendo la más común el agua común (o directamente sin especificaciones). Existen diversos tipos de aguas presentes. Véase Tabla 5.

Tabla 5: Tipos de agua y cantidad de veces presentes en los ensayos.

Agua	81
Agua ultrapura	3
Agua de mar	2
Agua salina	1
Agua desionizada	1
Agua desionizada pura	1
Agua dulce	1
Agua reciclada	1
Agua estancada de relaves	1
Agua de grifo	1

- Tipos de mineral

Para efectos de este trabajo de memoria, todos los ensayos considerados realizan flotación de cobre como mineral primario. Sin embargo, no todos aclaran qué tipo de mineral de cobre utilizan específicamente. Se muestran los minerales presentes en los ensayos. Véase Tabla 6.

Tabla 6: Tipos de mineral de cobre y cantidad de veces presentes en los ensayos.

Calcopirita	80
Relaves	4
Bornita	3
Calcosina	3
Malaquita	3
Covelina	2
Otros	3

Si un ensayo no posee especificación del mineral o minerales que presenta, entonces este parámetro se dejará en blanco. El caso de "Otros" solo se pondrá si al menos se presenta algún otro mineral, por ejemplo, "Calcopirita, Otros". En este caso, si bien Otros no es un mineral como tal, se hace de esta forma para que el testeo se relacione con minerales de Calcopirita, pero, a su vez, no lo tome como si fuera un número mineral presente, obteniéndose una proporción menor y más acertada.

- Tipo de celda

Solo se presentaron 3 tipos de celdas: Por columnas, mecánicas y neumáticas. Se aprecia la cantidad de veces en que se presentaron por testeo. Véase Tabla 7.

Tabla 7: Tipos de celdas de flotación y cantidad de veces presenten en los ensayos.

Mecánica	17
Columna	12
Neumática	1

4.1.7. Tamaño de partícula

Este parámetro es uno de los más complejos para tratar en el código, ya que el tamaño de partícula se puede presentar de muchas maneras, ya sea como un promedio o como un rango. Para simplificar esto, se realizará una clasificación binaria. Se tendrán dos categorías: Tamaño fino (menor a 200 μm) y tamaño grueso (mayor o igual a 200 μm), ambas excluyentes entre sí, y la clasificación se realizará de la siguiente manera:

- Si el tamaño de partícula es un valor promedio, se evaluará si dicho valor es mayor, igual o menor a 200 μm. Si es menor, se considerará como tamaño fino; en caso contrario, tamaño grueso.
- Si el tamaño se presenta entre dos valores X e Y, es decir, como un rango, se evaluará si ambos valores están sobre o bajo 200 μm. En caso de que el valor mayor esté sobre, y el valor menor esté por debajo, se realizará la clasificación con el promedio de estos dos valores.
- Si el tamaño de partícula se presenta como un rango donde solo se presenta un máximo, por ejemplo, que todas las partículas posean un tamaño menor a cierto valor, se

considerará partícula fina si este valor entregado no supera los 200 μm, en caso contrario, el tamaño de partícula no se inclinará ni por fina ni por gruesa, debido a la falta de información, por lo cual quedaría como un parámetro en blanco.

- Si se presenta como un valor pasante, es decir, por ejemplo, 80% de las partículas son menores a cierto valor, entonces solo se considerará como fino si este valor se presenta como 200 μm, además de que, para simplificar cálculos, se considerará ese valor como el máximo, ya que representa a la mayoría del muestreo. En caso contrario, el tamaño de partícula quedaría en blanco por falta de información.

4.1.8. Velocidad de gas

La velocidad del gas se presentará como velocidad o como caudal, dependiendo de la información entregada en cada paper. Algunas unidades presentes son: L/min, L/h, cm/s, entre otras.

4.2. Modelo en Python

El código implementa un sistema de inteligencia artificial orientado a la modelación y predicción de la recuperación de cobre en procesos de flotación, utilizando técnicas de aprendizaje supervisado basadas en árboles de decisión. Para ello, se emplea el lenguaje Python junto con librerías especializadas en análisis de datos, modelado estadístico y aprendizaje automático.

4.2.1. Librerías

En primer lugar, se importan bibliotecas fundamentales, tales como *pandas*, que permite la eficiente manipulación de datos tabulares, y *numpy*, que se centra en operaciones numéricas vectorizadas.

A estas se suman librerías del grupo *scikit-learn*, tales como *train_test_split* para la separación de conjuntos de entrenamiento y prueba, *RandomForestRegressor* como algoritmo principal de modelado no lineal, y *SimpleImputer* para tratar valores faltantes por medio de imputación estadística.

Adicionalmente, se incluyen librerías estándar como *argparse* para la gestión de argumentos por cada línea de comandos, *joblib* para la serialización y carga de modelos entrenados, *json* y *os* para el manejo de archivos y configuraciones persistentes, *matplotlib* para ver resultados gráficamente, y finalmente *datetime* y *math.ceil* para cálculos auxiliares.

4.2.2. Entrada de datos

Se define la función principal *FlotationAI*, la cual contiene las funciones esenciales del código desarrollado. La función *_init_* establece la configuración inicial del modelo, recibiendo como entrada la ruta del archivo Excel que contiene la base de datos de flotación, junto con otros archivos opcionales requeridos para su correcto funcionamiento. Durante la inicialización, se definen y preparan diversos atributos internos que permiten almacenar los datos originales y su versión procesada, así como las variables finales empleadas por el modelo, las categorías aprendidas a partir de la base de datos y del vocabulario del usuario, y los objetos asociados al modelo entrenado y al manejo de valores faltantes. Adicionalmente, se carga de forma automática un vocabulario personalizado desde un archivo JSON, lo que posibilita la incorporación de nuevos tipos de reactivos, aguas o minerales sin necesidad de modificar el código fuente. De este modo, la clase *FlotationAI* actúa como una estructura central que organiza el estado del sistema y asegura la coherencia entre el entrenamiento del modelo, el procesamiento de nuevas muestras y la generación de predicciones.

Como parte del código, se incorporan mecanismos para la gestión del estado del modelo y de los parámetros generados durante el proceso de aprendizaje, con el fin de garantizar la reproducibilidad de los resultados, evitar inconsistencias y permitir un re-entrenamiento completo cuando la base de datos es modificada. En este contexto, la función *reset_state* tiene como propósito reiniciar completamente la database de *FlotationAI*, eliminando de la memoria todos los datos y objetos generados en ejecuciones previas. En particular, se descartan tanto los datos originales cargados desde el archivo de entrada como su versión procesada, asegurando que cualquier cambio en la base de datos obligue a repetir todo el flujo de preprocesamiento. Finalmente, se eliminan también las referencias de valores faltantes y al modelo entrenado, evitando de esta manera el uso de configuraciones que no sean coherentes con una nueva base de datos. Todo esto permitirá ir actualizando la base de datos sin que el código considere estudios previos.

De forma complementaria, la función *delete_cached_artifacts* se encarga de eliminar los archivos persistentes generados en ejecuciones anteriores, los cuales almacenan modelos, transformadores y metadatos. Esta función recorre un conjunto de archivos clave, como el modelo entrenado, el imputador, la definición de las columnas de entrada y el diccionario maestro de tipos, además de verificar su existencia antes de eliminarlos. El proceso se realiza de forma controlada, informando al usuario sobre la eliminación exitosa de los archivos y emitiendo advertencias en caso de error. La distinción entre el reinicio del estado en memoria y la eliminación de artefactos persistentes permite un manejo adecuado del ciclo del modelo, facilitando la actualización del sistema y evitando el uso de configuraciones obsoletas frente a cambios en los datos o en el preprocesamiento.

4.2.3. Carga y reproceso de datos

La función *load_data* constituye el punto de entrada para la incorporación de datos experimentales al sistema de inteligencia artificial de flotación, siendo responsable de la lectura del archivo de entrada y de la inicialización coherente del estado interno del objeto. Este método permite al usuario especificar opcionalmente una hoja del archivo Excel y decidir si el estado previo del modelo debe reiniciarse antes de la carga. En caso de que el parámetro *reset* se establezca como verdadero, el sistema invoca los métodos *reset_state* y *delete_cached_artifacts*, asegurando la eliminación de cualquier rastro de ejecuciones anteriores, tanto en memoria como en disco, antes de iniciar un nuevo ciclo de procesamiento. Este procedimiento es importante al momento de incorporar nuevos datos.

El método verifica la existencia y validez de la ruta del archivo de datos, considerando distintos atributos posibles para asegurar la compatibilidad. Es importante que la base de datos de Excel se encuentre en la misma carpeta que el resto del código de Python para que así sea capaz de encontrar la ruta.

Una vez validada la ruta, el archivo Excel se carga utilizando la abstracción *ExcelFile* de la biblioteca pandas, lo que permite acceder de manera eficiente a una o varias hojas sin recargar el archivo completo. Según la configuración del usuario, se carga una hoja específica o la hoja por defecto del archivo. Como resultado, se obtiene una database que representa los datos originales tal como fueron definidos en el archivo Excel.

Posteriormente, se aplica una etapa de limpieza básica orientada a normalizar los valores faltantes y eliminar información irrelevante, más específicamente, se tratan las celdas vacías de la base de datos. En esta fase, los valores nulos se convierten explícitamente en NaN de *NumPy* para asegurar compatibilidad con los procesos de imputación, y se realiza una inferencia de tipos de datos para corregir inconsistencias comunes en archivos Excel. También se eliminan las columnas que contienen exclusivamente valores nulos, reduciendo posibilidades de errores y evitando interferencias en las etapas posteriores de procesamiento.

Finalmente, la base de datos resultante se almacena en los atributos internos del modelo, conservándose una versión activa y otra versión copia que permiten preservar el estado original de los datos para análisis o reprocesamiento posterior. El método concluye informando al usuario sobre la carga exitosa del archivo y el número de filas disponibles.

Por otra parte, la función *preprocess_data* indica la transición entre la carga de los datos y su preparación para el modelado predictivo, transformando la información original en una representación numérica adecuada para el aprendizaje automático. El método verifica previamente que los datos hayan sido cargados, asegurando una correcta ejecución, y luego llama a la función *_process_df*. Este proceso realiza la limpieza y normalización de las variables, almacenando el resultado en *df_processed*, que constituye la base definitiva para el entrenamiento del modelo.

4.2.4. Predicción del modelo

Una vez procesados los datos, el código define el conjunto de variables predictoras. Si la columna Recuperación de Cu está presente, esta se excluye del conjunto de características, pues es lo que se intentará predecir; en caso contrario, se asume que todas las columnas corresponden a variables de entrada, lo que permite utilizar el método en casos de predicción.

Por otro lado, el método *train_model* constituye el núcleo del aprendizaje automático del modelo, siendo responsable del entrenamiento y la evaluación de la predicción de recuperación de cobre. La función verifica que el procesamiento inicial haya sido realizado y que la variable objetivo (recuperación) esté presente. Posteriormente, se utilizan únicamente los datos que contienen valores reales de recuperación y se establece un mínimo

de datos para el entrenamiento, con el fin de evitar modelos estadísticamente inestables. En este caso, el mínimo será de 10 ensayos para construir una database.

A continuación, se construyen las matrices de entrada y salida del modelo. La variable objetivo se convierte a formato numérico, asignando valores no interpretables o vacíos como NaN. El sistema detecta automáticamente la escala de la recuperación y, en caso de estar expresada de forma fraccional, la convierte internamente a porcentaje. Este mecanismo permite integrar datos de distintas fuentes sin requerir una estandarización manual previa. Con esto, el usuario puede ingresar, en la base de datos, el valor de recuperación como porcentaje o decimal como estime conveniente y el código lo interpretará de igual forma, haciendo la transformación interna.

De forma paralela, las variables predictoras se limpian y convierten a formato numérico, eliminando columnas completamente vacías. Posteriormente, se conservan solo las observaciones con valores conocidos de recuperación. Finalmente, se define y fija el conjunto de variables utilizadas por el modelo para asegurar una buena reproducibilidad en las predicciones posteriores.

El modelo utilizado corresponde a un Random Forest con 200 árboles, y su comando es *RandomForestRegressor*. Esto significa que el modelo predecirá mediante una serie de árboles de decisión para, finalmente, promediar todas esas predicciones y así entregar un valor de recuperación.

Avanzando en el código, la función *predict_from_dict* implementa la parte final del modelo, el que permitirá predecir la recuperación de cobre para nuevos casos ingresados por el usuario. Este método asegura que los datos de entrada sean procesados bajo las mismas reglas y transformaciones utilizadas durante el entrenamiento.

Se verifica en el siguiente bloque que los componentes esenciales del código se encuentren correctamente inicializados, incluyendo la lectura de la base de datos, el imputador y el diccionario maestro. Esto valida que la ejecución haya sido realizada correctamente antes de efectuar una predicción, y en caso contrario genera un error explícito.

La siguiente función, *model_features*, corresponde al conjunto de variables utilizadas durante el entrenamiento. Esto garantiza que la predicción se realice con la misma estructura de entrada que el modelo aprendió y, en caso contrario, el método interrumpe la ejecución, indicando que el modelo no se encuentra correctamente entrenado. Esto consta solamente de un añadido de seguridad para evitar usar el código en condiciones que no son ideales.

Cuando ya estén hechas las validaciones, los parámetros de entrada se convierten en un data frame de una sola fila. Este se procesa con el aprendizaje de tipos desactivado, asegurando que se utilicen únicamente las categorías y transformaciones definidas durante el entrenamiento y evitando modificaciones en la estructura interna del modelo.

Posteriormente, el data frame procesado se convierte completamente a formato numérico para que coincida exactamente con las variables utilizadas durante el entrenamiento. Las columnas faltantes se incorporan como valores nulos y las columnas adicionales se descartan, asegurando que la estructura de entrada sea compatible con el modelo y evitando errores en la predicción.

El valor predicho se convierte a tipo numérico y, si corresponde, se ajusta automáticamente a porcentaje. Finalmente, el método devuelve la recuperación estimada redondeada a dos decimales, entregando un resultado claro y consistente con la variable objetivo del sistema. En conjunto, el modelo refleja el enfoque del sistema hacia predicciones reproducibles y coherentes, incluso frente a datos incompletos, preservando la integridad del modelo entrenado.

Posteriormente, se definen funciones auxiliares para gestionar de forma segura la entrada de datos del usuario y distinguir entre valores omitidos y valores efectivamente ingresados. Esta distinción resulta clave para respetar el formato utilizando y evita autocompletar datos faltantes en la database original.

Se destaca que el código opera mediante un bucle continuo que permite realizar múltiples predicciones consecutivas. En cada iteración, el código solicita los parámetros de un nuevo experimento de flotación, los cuales se almacenan en un archivo que luego es procesado por el mismo flujo utilizado durante el entrenamiento.

En cada iteración, se solicitan al usuario variables numéricas básicas del proceso, como pH, velocidad de gas, tiempo de flotación y tiempo de acondicionamiento, así como información sobre el tamaño de partícula y su unidad. El sistema permite una representación flexible de estos valores, los cuales son interpretados posteriormente por los mecanismos internos de procesamiento.

Posteriormente, se define una función para gestionar los distintos grupos de reactivos, utilizando un menú interactivo basado en los parámetros aprendidos por el modelo gracias a la database o a predicciones previas. Este mecanismo permite seleccionar múltiples reactivos y definir sus dosis y unidades de forma sencilla, seleccionando números en un listado completo de opciones.

Para las categorías como tipo de agua, mineral o celda, se emplea una función que presenta opciones basadas en el vocabulario del modelo y permite selecciones múltiples. En el caso de tipo de agua y tipo de mineral, también es posible seleccionar más de una opción, dependiendo de lo que fue utilizado en casa ensayo.

Antes de generar la predicción, el modelo convierte las variables numéricas (pH, velocidad de gas y tiempos) a su formato correspondiente, gestionando posibles errores de formato. Este paso reduce fallos en etapas posteriores sin interrumpir la interacción del código con el usuario.

Una vez que el usuario entrega todos los parámetros de un testeo, el código invoca la función de predicción utilizando el mismo esquema aplicado durante el entrenamiento y presenta en la pantalla la recuperación estimada en formato porcentual, facilitando de ese modo la interpretación del resultado. Posterior a eso, la pantalla interactiva permite al usuario ingresar el valor real de recuperación cuando esté disponible, registrándolo junto con la predicción generada. En ausencia de este valor, se almacena un dato nulo, dándole a entender al código que no se posee un valor de recuperación real para esos parámetros estudiados.

Entonces, el modelo incorpora un mecanismo de retroalimentación con reentrenamiento inmediato. Cuando se dispone de un valor real de recuperación, este se utiliza como etiqueta; en caso contrario, se emplea el valor de la predicción como una etiqueta provisoria. En ambos

escenarios, el nuevo experimento se integra al conjunto de datos y el modelo se actualiza automáticamente, considerando ese nuevo ensayo como parte de la database ingresada originalmente. El proceso interactivo finaliza cuando el usuario decide terminar la sesión, ya sea porque ya no desea predecir nada más o por los motivos que se estimen convenientes. En caso de haber registrado valores reales en algunos de los testeos, el sistema genera un pequeño gráfico comparativo en formato png entre predicciones y observaciones, y finalmente guarda los nuevos términos o parámetros actualizados, asegurando la persistencia de los aprendizajes para ejecuciones futuras.

El código prosigue con *_append_feedback_and_retrain*, cuyo propósito principal es registrar cada nuevo testeo de flotación, junto con su valor de recuperación, para así integrarlo a la database previa y, cuando las condiciones lo permiten, reentrenar inmediatamente el modelo incorporando esta nueva información. Esta función recibe tres argumentos que permiten este registro*: sample_original_format*, que preserva el testeo en el formato original del Excel sin preprocesamiento; *y_value*, que corresponde a la recuperación utilizada como etiqueta, ya sea real o predicha (dependiendo de que si el testeo poseía o no una recuperación real); y *source*, que indica la especificación de dicha etiqueta (real o predicción). Esta diferenciación facilita el análisis posterior del aprendizaje incremental del modelo. La sección construye una nueva fila a partir del dataframe original, incorporando la variable objetivo "Recuperación Cu (%)" junto con metadatos de control *_source* y *timestamp*. Si bien estos campos no participan en el entrenamiento, funcionan para ayudar al posterior análisis del modelo.

A continuación, el modelo actualiza la base de datos en la memoria (*self.df_original*), inicializándola con la nueva fila de datos. Con la base de datos actualizada, el código reprocesa los datos, permitiendo de este modo aprender nuevos tipos de valores para cada parámetro y ese esa manera actualizar los catálogos maestros que contienen información teórica.

La función intenta reentrenar el modelo mediante *train_model()*, informando al usuario si la actualización con la base aumentada fue exitosa. En caso de fallo, el error se registra sin perder el feedback previo, que permanece almacenado en memoria y en el Excel aumentado para futuros entrenamientos, cerrando el ciclo de predicción, retroalimentación y aprendizaje.

Esto permite que el código pueda seguir funcionando a pesar de que no se logró guardar nueva información en la base de datos.

4.2.5. Utilidades internas

Esta sección del código permite hacer transformaciones o enlistar los parámetros necesarios para que cada variable sea comparable con las demás.

Se utilizan las librerías importadas y se definen utilidades internas para normalizar unidades químicas y de gas. Estas funciones convierten las magnitudes a familias coherentes (por ejemplo, g/t o kg/t para reactivos, m/s o m^3/s para gas), para así, de esta manera, realizar mejores comparaciones entre los parámetros de la base de datos.

Además, el siguiente bloque del código se centra en interpretar y normalizar el tamaño de partícula, un parámetro crítico en flotación que se presenta de distintas formas en papers y, por ende, en la base de datos, por lo cual se debe reclasificar de manera binaria (fina y gruesa) para simplificación de los cálculos, siendo 0 para gruesa y 1 para fina, manteniéndose NaN cuando el dato no está disponible. La función *_to_um* complementa este procesamiento del tamaño de partícula. Su principal función es convertir todos los valores a micrómetros (µm), que corresponde a la unidad estándar del tamaño de partícula en el código. Cuando la unidad declarada es milímetros (mm), se realiza una transformación de unidad, multiplicando el valor por 1000. En los casos en que la unidad no se indique en el código, el modelo asume de forma automática que el tamaño ya se encuentra expresado en micrómetros. En conjunto, esta función asegura que la clase reciba una representación numérica del tamaño de partícula, independientemente del formato en que haya sido reportado originalmente este dato.

Posteriormente, el código procede a ignorar columnas de carácter bibliográfico o de parámetros que finalmente no se utilizaron, como el tamaño de las burbujas o el tipo de gas, pues, como se explicó con anterioridad, no se consideraron esos valores para predecir la recuperación, por lo que su tratamiento en el código generaría ruido innecesario en los resultados.

A continuación, se procesa la información asociada al gas de flotación, diferenciando explícitamente entre velocidad y caudal. Para ello, se generan columnas separadas que

representan la velocidad en m/s y el caudal en m^3/s, que son las unidades del sistema internacional, aunque no las más comunes para tratar al gas en la flotación. Además, se incorporan indicadores que señalan si la unidad original correspondía a un caudal, si la unidad no pudo ser identificada o si fue necesario asumir una velocidad en m/s debido a que el usuario no proporcionó información sobre una unidad. Este parámetro es uno de los más ambiguos de tratar, pues no es posible relacionar los datos entregados como velocidad con los entregados como caudal debido a la falta de información. Se tendría que disponer de un área para cada celda en la database para poder hacer una transformación y una comparativa de todos los datos.

Finalmente, se construye un catálogo maestro que reúne todos los tipos observados en la base de datos, integrándolos además con los vocabularios previamente guardados por el usuario en el archivo *JSON*. A partir de este catálogo, se generan de manera segura columnas binarias que indican la presencia de cada categoría. Esto permite tener un registro de todos los tipos de parámetros ingresados en la database, como reactivos, minerales, etc., y se da la opción al usuario de elegir en nuevas predicciones entre esas opciones o si añadir nuevas.

La sección de los reactivos es similar a la de la velocidad del gas, donde no solo es importante el tipo de reactivo utilizado, sino también la dosis y su unidad. Para ello, el código define unos grupos que organizan cómo se relacionan cuatro elementos en cada familia de reactivos: la columna que contiene el nombre del reactivo, la columna asociada a la dosis, la unidad correspondiente a dicha dosis y el tipo de reactivo (colector, depresante, espumante u otros). Esta forma de organización permite que el código evite la duplicación de datos. Este procedimiento es similar al utilizado para el tipo de agua, mineral y celda, pero solo considera una columna.

Para cada grupo de reactivos, el código obtiene el conjunto de tipos conocidos desde el catálogo maestro y genera con esto variables binarias. Estas variables indican de forma simple si un reactivo específico estuvo o no presente en cada ensayo, independientemente de la dosis aplicada. De esta manera, el modelo permite que el código capture información cualitativa de los reactivos incluso cuando la información cuantitativa es incompleta o no similar, lo cual permite más caminos posibles en los distintos árboles de decisión.

A continuación, se generan varias variables relacionadas a las dosis de los reactivos. Entre ellas se incluyen la dosis total normalizada a unidades comparables, como g/t o kg/t, así como indicadores que registran situaciones relevantes. Estas situaciones incluyen, por ejemplo, el uso de una única dosis para varios reactivos (ya que se trataba con una mezcla como reactivo), inconsistencias entre el número de reactivos, dosis y unidades declaradas, la presencia de dosis sin un tipo asociado o unidades que no correspondan. Esta información, sin embargo, no se considera como si fueran errores, sino variables se obtienen debido a incertidumbre y ambigüedad de los datos experimentales, permitiendo que el modelo aprenda a partir de esta información en lugar de descartarla, por lo cual sería responsabilidad de la database cargada este tipo de situación y de los datos ingresados para predecir posteriormente.

Después de esto, cada dosis se intenta convertir a un valor numérico y se clasifica según su tipo dimensional, el cual, para efectos de flotación, debería ser masa por tonelada, aplicando los factores de conversión necesarios para llevar todas las dosis a la unidad estándar g/t. Así se logra que dosis reportadas en distintas unidades sean comparables entre sí.

El resultado es un data frame que combina variables cualitativas y variables cuantitativas ya normalizadas en una misma unidad, reflejando las diferencias de los datos de flotación, pero expresada en un formato que puede ser utilizado eficazmente por modelo el para una correcta comparación de los datos.

4.2.6. Líneas de comandos

La interfaz de línea de comandos cumple el rol de coordinar el funcionamiento general del código, permitiendo que el modelo se utilice como una herramienta configurable sin necesidad de modificar el código fuente. Mediante el uso de *argparse*, se definen los parámetros principales del proceso, como la ruta del archivo Excel, la hoja de datos y algunos ajustes experimentales básicos. Esta capa facilita la repetición de experimentos y la comparación de resultados bajo distintas configuraciones. Además, incorpora opciones de control que permiten eliminar el conocimiento acumulado de ejecuciones anteriores o iniciar una corrida (run) completamente aislada. Esto ayuda a evitar mezclas no deseadas entre corridas y a evaluar el comportamiento del modelo de forma más ordenada, dependiendo de las necesidades del usuario.

En la etapa final del flujo de ejecución, el sistema crea una instancia utilizando los parámetros previamente definidos desde la línea de comandos. En esta inicialización se especifican elementos clave como la ubicación de los datos, los archivos donde se guarda el vocabulario del usuario y el aprendizaje acumulado, además del directorio de salida de resultados.

En la siguiente sección de código se ejecuta el proceso de carga de datos, asegurando un reinicio del sistema para evitar que queden registros de ejecuciones previas. Este paso funciona como una verificación inicial del entorno: Si los datos no pueden cargarse correctamente, el programa se detiene de manera explícita. Así se previene que el modelo continúe con entrenamientos o predicciones basados en información incompleta o dañada.

Finalmente, una vez verificada la carga de los datos, el código transforma la información original en una forma numérica ordenada y compatible con el modelo de aprendizaje automático. En esta etapa se definen las variables de entrada, se normalizan las unidades y se procesan las variables categóricas, dejando el conjunto de datos preparado para las etapas posteriores de entrenamiento o predicción. De esta manera, el preprocesamiento actúa como el puente entre los datos experimentales originales y el modelo que realizará las estimaciones.

4.3. Predicción y comparativa con valores existentes

A continuación, se utilizará el código para predecir valores de recuperación en testeos que poseen su recuperación real, cerrando con una comparativa de ambas recuperaciones.

Se inicia con una base de datos que servirán como una primera entrada en el modelo, mientras que los demás testeos se usarán para predecir sus recuperaciones.

Se hicieron unas pruebas iniciales de predicción con ensayos aleatorios como database, dando resultados no cercanos a la realidad. Esto se debe a la gran diferencia entre los parámetros que se están comparando entre sí y la escasa cantidad de datos en total. Como el modelo trabaja con lo que tiene, la gran dispersión entre parámetros y la falta de una buena cantidad de testeos es algo que afecta a los resultados finales del código. Esto no significa que el modelo no funcione correctamente, sino que hace falta una database más robusta para que entregue resultados más satisfactorios.

Debido a esta condición, se evaluó una mejor forma de realizar una buena comparativa. Considerando que los testeos 4 al 71 son muy similares entre sí, al provenir de un mismo documento bibliográfico, variando solo en valores de pH y reactivo adicional con su respectiva dosis, se decidió que estos serían los ensayos a los que se les predeciría su recuperación y se compararía con su valor real. Al haber poca dispersión entre estos ensayos, se espera que los resultados entregados por el modelo serán más exactos que antes.

Se muestran los parámetros fijos que poseen los ensayos del 4 al 71. Véase Tabla 8.

Tabla 8: Parámetros constantes de los ensayos 4 al 71.

Tipo de colector	SIPX y DBS
Tipo de espumante	MIBC y Nasfroth
Tipo de Agua	Agua
Tipo de minera	Calcopirita
Tipo de celda	Columna
Velocidad gas	1,5 cm/s
Tiempo de flotación	8 minutos
Tiempo de acondicionamiento	3 minutos

Mientras tanto, los parámetros que varían son:

- pH: Entre 10 y 12.
- Otros reactivos: SiO_2, TiO_2, α-Fe_2O y γ-Al_2O_3.
- Dosis otros reactivos: Entre 4 y 6 kg/t.

Con esto, se tienen 68 testeos en total con parámetros similares, diversas combinaciones y diferentes recuperaciones de cobre. En contraste, los otros 32 datos se utilizarán como base de datos de entrada, lo cual es congruente con el modelo, considerando que pide como mínimo 10 datos de entrada.

Los 68 ensayos fueron ordenados de manera aleatoria en el siguiente orden: 2, 41, 49, 17, 21, 18, 19, 6, 25, 46, 29, 22, 38, 31, 16, 47, 27, 62, 53, 67, 37, 48, 64, 30, 14, 58, 40, 42, 52, 51, 8, 36, 39, 5, 13, 33, 57, 61, 35, 54, 24, 20, 28, 26, 65, 34, 10, 7, 11, 59, 66, 15, 12, 9, 3, 4, 43, 68, 63, 56, 45, 23, 60, 32, 55, 50 y 44. Una vez que el código comenzó a correr, se ingresaron los parámetros del primer testeo (en este nuevo orden aleatorio) y el modelo logró predecir

una recuperación. Posterior a ello, el código pregunta al usuario si posee el valor real de este testeo, el cual debe ser ingresado para que el modelo adquiera este nuevo testeo a su base de datos con el valor correcto. Finalizado esto, el código preguntará al usuario si se desea predecir un nuevo valor de recuperación, en este caso se señala que sí y se prosigue con el siguiente ensayo, y así sucesivamente hasta llegar al último.

Se muestran las predicciones de recuperación, realizándose también una comparativa con las recuperaciones reales de dichos testeos. Véase Figura 10.

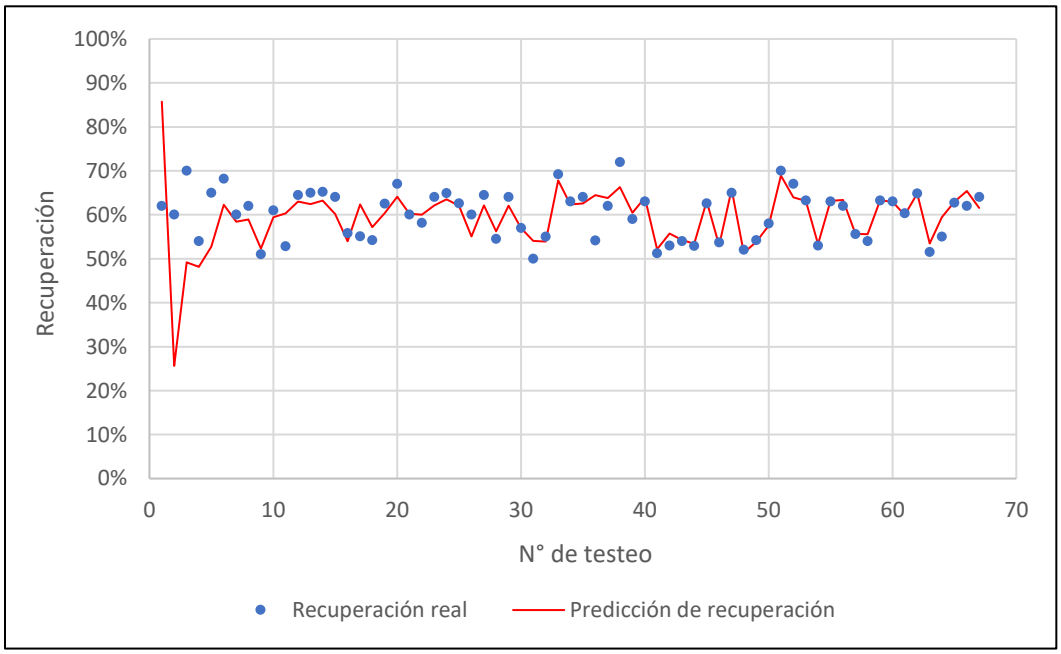


Figura 10: Comparativa de recuperación real y su predicción.

Analizando el gráfico, se puede observar que, en un inicio, las predicciones se alejan de los valores reales, mientras que, a medida que se avanza con los ensayos, los valores de predicción comienzan a ser mucho más cercanos a los reales. Esto coincide con la idea de que el modelo va aprendiendo a medida que adquiere más ensayos a su base de datos. Además, que sean solo tres los parámetros que van variando en un rango pequeño es de ayuda para este tipo de experimento.

Se observa además el único caso donde la recuperación real coincide con la predicción del código. Véase Figura 11.

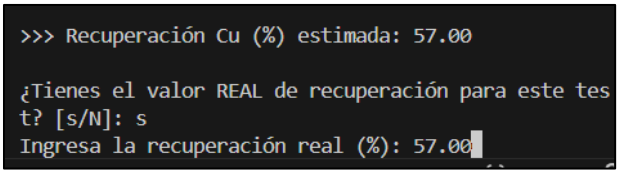


Figura 11: Predicción exacta de un parámetro.

Se muestra además una comparativa de Recuperación vs. pH de los ensayos que se quisieron predecir, basado en la recuperación real y en la predicción. Véase Figura 12. En el gráfico se construyó una curva promedio de recuperación real y predicción para cada pH. Se pueden apreciar los valores asociados a cada curva. Véase Tabla 9.

Para el pH 10, el promedio de recuperaciones coincide para una recuperación de dos decimales, y para los otros dos pHs, los valores siguen siendo cercanos entre sí, lo cual apoya la idea de que los resultados de predicción de recuperación son aceptables y cercanos a la realidad. Esto también se demuestra visualmente, pues la curva de predicción se comporta de forma realista como una verdadera curva de recuperación de cobre vs. pH.

Tabla 9: Recuperaciones promedio para cada pH.

pH	Recuperación	Predicción
10	54,08%	54,08%
11	62,57%	61,17%
12	63,01%	62,41%

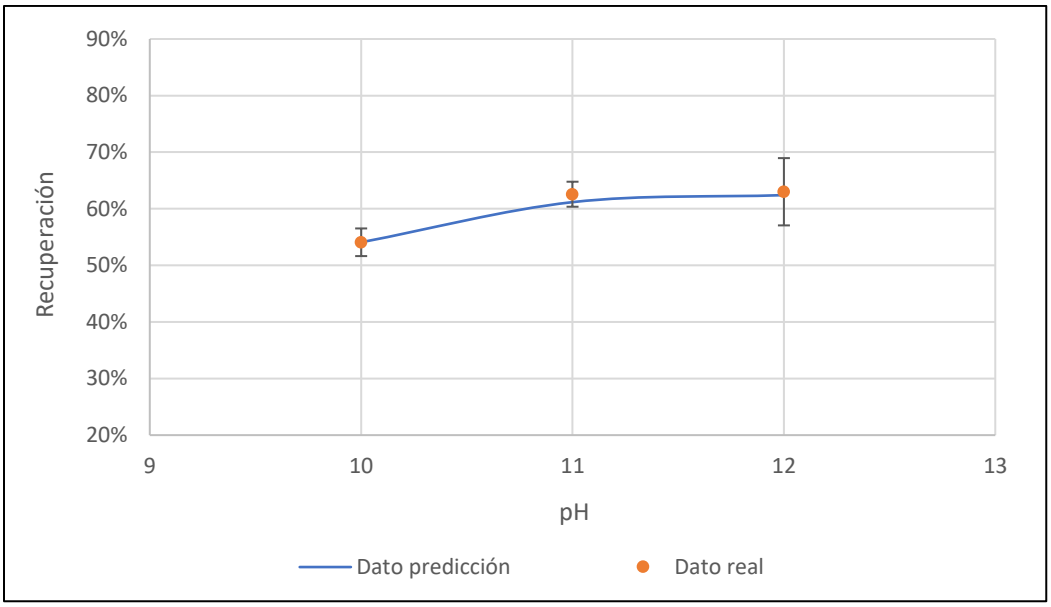


Figura 13: Gráfico recuperación real vs pH, que incluye barra de error por desviación estándar, y predicción de recuperación vs pH.

4.4. Predicción de nuevos ensayos con combinaciones aleatorias

Finalmente se realizaron ensayos con combinaciones aleatorias, utilizando parámetros entregados en la database (Anexo C).

El testeo n°1 posee un valor eficiente de flotación debido a su combinación, la cual no es muy diferente de otros ensayos ya presenten en la database. Véase Tabla 10.

A su vez, los testeos 4, 5 y 6 poseen valores altos debido a que utilizan colectores que están presenten en gran parte de los ensayos de la database, lo cual genera varios caminos en el árbol de decisión y, por ende, más predicciones útiles en el Random Forest.

Finalmente, los demás ensayos presentan valores bajos de recuperación, lo cual se puede asociar a que utilizan parámetros que se presentan pocas veces en la base de datos, además de presentar combinación de parámetros que no son del todo comunes en esta misma database.

Tabla 10: Resultados de predicción de ensayos con combinaciones aleatorias.

Test	Recuperación Cu
1	54,66%
2	1,17%
3	1,80%
4	80,75%
5	56,86%
6	86,07%
7	9,27%
8	11,49%

Debido a su programación general como Randon Forest de regresión, el modelo entrega valores ineficientes cuando un parámetro no se encuentra varias veces repetido en la database o los parámetros no se presentan combinados de distintas maneras, lo cual es uno de los puntos débiles del código. Esto indica que el código depende de tener una database robusta.

4.5. Análisis general del modelo

4.5.1. Ventajas generales

El modelo presenta una forma completa de predicción, tomando en consideración una serie de parámetros para una mayor complejidad a la hora de predecir y tomar caminos en los diversos árboles de decisión en el Random Forest.

Además, se construye una manera de aprendizaje en el tiempo, adquiriendo nuevos ensayos a la database y comparando valores de predicción con valores reales de recuperación, lo que también permite mejorar la exactitud a medida que se siguen añadiendo nuevos ensayos a la base de datos.

Tomando en cuenta los resultados, el código de Python ofrece resultados cercanos a la realidad cuando el código se entrena con ensayos similares entre sí y se intenta predecir algo parecido a esos datos. Esto es útil para experimentos de laboratorio en los cuales solo se desean variar pocos parámetros para evaluar su efectividad en la flotación.

Como ventaja final, es importante destacar la capacidad de aprendizaje del modelo, el cual va adquiriendo nuevo conocimiento cada vez que se intenta predecir un nuevo testeo, además

de ofrecer la opción de colocar la recuperación real para corregir al modelo y que su database sea lo más exacta posible.

4.5.2. Desventajas generales

El mayor problema que presenta este modelo de predicción es que depende mucho de una base de datos robusta que no varíe demasiado entre sus parámetros, o al menos que cada muestra similar de parámetros cuente con varios ensayos dentro del código. Si se desea predecir una recuperación, los resultados serán satisfactorios si en la database existe parámetros que se repiten constantemente y coinciden con los parámetros ingresados. Sin embargo, los resultados pueden ser deficientes si los parámetros no se presentan una cantidad de veces considerable en la database, llegando a predecir valores de recuperación muy alejados de la realidad.

Otro riesgo que se puede presentar es intentar predecir valores de recuperación sin tener el valor real, pues el modelo, para efectos de este trabajo de memoria, va aprendiendo de esto, lo que provoca alimentarse de valores que pueden no ser exactos, afectando futuros análisis. Es por eso que se recomienda predecir cuando se tenga una base de datos robusta y similar a lo que se desea predecir.

Finalmente, hay que considerar que este modelo se basa puramente en su código, es decir, no considera efectos adversos ante ciertas combinaciones de parámetros. Por ejemplo, si dos reactivos no deberían mezclarse entre sí porque puede afectar a la recuperación, el modelo no lo va a saber y solo basará sus resultados en la database previa y su aprendizaje. Es importante de considerar a la hora de realizar una predicción de combinaciones aleatorias.

5. CONCLUSIONES

Fue posible reunir los suficientes testeos de flotación de cobre para poder construir una database de entrada para el modelo predictivo, los cuales varían en sus diferentes parámetros, permitiendo así tener una amplia base de datos.

Se logró construir un modelo capaz de entregar predicciones de recuperación para flotación de cobre mediante un código de Python, el cual acepta los parámetros de entrada mediante un archivo Excel.

Los resultados muestran que el modelo es capaz de entregar predicciones, pero no siempre entrega valores cercanos a la realidad.

El modelo fue capaz de predecir testeos que eran similares entre sí, variando solo unos pocos parámetros, además de que se contaba con la recuperación real de cada uno, dándole al código la oportunidad de aprender y re-entrenarse. A medida que se iba prediciendo y re-entrenando, los resultados iban siendo más exactos.

Sin embargo, cuando se intentó predecir ciertos ensayos que fueron creados con parámetros aleatorios y que no se repetían constantemente en la base de datos de entrada, los resultados fueron deficientes, dando como resultados recuperaciones bajas como 1,17%, 1,80% y 11,49%, lo cual se aleja de cualquier recuperación entregada previamente como entrada, siendo 47,02% la más baja. Esto demostró que el modelo tiene su punto bajo: Depende mucho de los datos de entrada, por lo cual, si los datos no están ampliamente presentes ahí, los resultados no serán buenos.

Por ello, el modelo logra predecir cuando la base de datos es robusta y con parámetros similares entre sí, lo cual es útil para casos donde la mayoría de los parámetros de la flotación son fijos y se decida estudiar algo en particular que se varíe, como un reactivo en específico o alguna dosis.

Lograr predecir previamente recuperaciones de este modo antes de analizar y obtener resultados reales es una forma eficaz de optimizar tiempos y, a su vez, el proceso de flotación.

REFERENCIAS

- Belcic. *What is classification in machine learning?*
- Brest et al. (2021). *Statistical investigation of flotation parameters for copper recovery from sulfide flotation tailings.*
- Chernousenko & Kameneva (2021). *The Use of Tecflote Family Collectors in Copper–Nickel Ore Flotation.*
- CODELCO (2019). *FLOTACIÓN: "Burbujas de cobre".* Codelco Educa.
- Cruz (2021). *Current Status of the Effect of Seawater Ions on Copper Flotation: Difficulties, Opportunities, and Industrial Experience.*
- Faouzi et al. (2024). *Predictive Geometallurgical Modeling for Flotation Performance in Mixed Copper Ores Using Discriminatory Methods.*
- Gabasiane et al. (2024). *Reclamation of iron and copper from BCL slag in Botswana.*
- Gizatullina et al. (2023). *About the use of R-66 reagent in the technology of flotation enrichment of mixed copper.*
- Hassanzadeh et al. (2024). *ImhoflotTM Flotation Cell Performance in Mini-Pilot and Industrial Scales on the Acacia Copper Ore.*
- Ibrahim et al. (2021). *Optimization of Process Parameters for Enhanced Up-gradation of Qilla Saifullah Copper ore through Froth Floatation Technique.*
- James et al. (2013). *An Introduction to Statistical Learning with Applications in R*.
- Kavlakoglu. *What is random forest?* IMB.
- Keita (2024). *Classification in Machine Learning: An Introduction.*
- *King (1982). Flotation of fine particles.*
- Li et at. (2021). *Physicochemical, Mineralogical Liberation Characteristics, and Direct Recovery of Copper and Iron from Copper Electric Furnace Slag.*
- Liao et al. (2022). *Application of macromolecular organic polymer S-7261A in arsenic removal by flotation of refractory mixed copper ore.*
- Liu (2013). *A review of the effect of water quality on flotation.*
- Lu et al. (2023). *A novel method for improving sulfidization xanthate flotation of malachite: Copper–ammonium synergistic activation.*
- Metso. *Actualización para controlar el aire de flotación.*

- Nasirimoghaddam et al. (2020). *Assessment of pH-responsive nanoparticles performance on laboratory column flotation cell applying a real ore feed.*
- Özçelik & Ekmekçi (2022). *Reducing Negative Effects of Oxidation on Flotation of Complex Cu–Zn Sulfide Ores.*
- Peng et al. (2024). *Sodium trithiocarbonate as an activator for malachite xanthate flotation: Experimental study and practical application.*
- Qin et al. (2024). *Enhanced recovery of low-grade copper ore and associated precious metals from iron tailings: A case study in China.*
- Rusell & Norvig (2004). *INTELIGENCIA ARTIFICIAL UN ENFOQUE MODERNO* (2da edición).
- Semushkina et al. (2022). *Processing of mature copper tailings from concentration plant using a composite reagent.*
- Semushkina et al. (2022). *Improving the Copper-Molybdenum Ores Flotation Technology Using a Combined Collecting Agent.*
- Semushkina et al. (2023). *Flotation processing of copper-containing technogenic raw materials using a composite flotation reagent.*
- Seyrankaya et al. (2023). *Beneficiation Of Artvin-Cerattepe copper-zinc ore by flotation.*
- Shen (2022). *Microbubble and nanobubble-based gas flotation for oily wastewater treatment: a review.*
- Soufiabadi et al. (2023). *Effect of different process water sources on rougher flotation efficiency of a copper ore: A case study at Sarcheshmeh Copper Complex.*
- Štirbanović et al. (2022). *Application of Thionocarbamates in Copper Slag Flotation.*
- Sun (2024). *Efficient recovery of copper from copper smelting slag by gravity separation combined with flotation.*
- Tanaka et al (2021). *Mineralogical Prediction on the Flotation Behavior of Copper and Molybdenum Minerals from Blended Cu–Mo Ores in Seawater.*
- Tardivon (2022). *Random Forest: Bosque aleatorio. Definición y funcionamiento.*
- Wang et al. (2020). *Regulation of bubble size in flotation: A review.*
- Wang et al. (2024). Amorphous silica effects on copper flotation: A kinetic and selectivity investigation.

- Wills & Finch (2016). *Wills' mineral processing technology: An introduction to the practical aspects of ore treatment and mineral recovery* (8th edition).
- Wisnu et al. (2024*). Selective Flotation of Copper Concentrates Containing Arsenic Minerals Using Potassium Amyl Xanthate and Oxidation Treatment.*
- Xiao et al. (2024). *Flotation separation mechanism of malachite from calcite using the pentyl xanthate as the collector.*
- Yianatos & Vinnett (2015). *Flotación de minerales: Fundamentos, tecnología y aplicación*.
- Zeng et al. (2023). *Effect of preoxidation on copper flotation from copper-lead mixed concentrate.*
- Zhai et al. (2022). *Mineralogical characteristics of copper smelting slag affecting the synchronous flotation enrichment of copper and arsenic.*
- Zhao et al. (2022). *Enhanced sulfidization flotation mechanism of smithsonite in the synergistic activation system of copper–ammonium species.*

ANEXOS

Anexo A: Código de Python para predicción de flotación.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
import argparse
import joblib
import json
import os
from math import ceil
import matplotlib.pyplot as plt
from datetime import datetime


class FlotationAI:

    def __init__(self, input_file: str, user_vocab_path: str =
"./user_vocab.json",
                 outdir: str = "./outputs", augmented_path: str |
None = None):
        self.input_file = input_file
        self.user_vocab_path = user_vocab_path
        self.outdir = outdir
        self.augmented_path = augmented_path or
os.path.join(outdir, "augmented_feedback.xlsx")
        self.df_original = None
        self.df_processed = None
        self.feature_columns = None
        self.tipos_maestro = None
        self.model = None
        self.imputer = None
        self._user_vocab = self._load_user_vocab()

    def reset_state(self):
        """Borra todo lo derivado/aprendido para forzar un
reprocesado completo."""
        self.df_raw = None
        self.df_processed = None

        self.feature_columns = None
        self.tipos_maestro = None
```

```
        self.imputer = None
        self.model = None
    def delete_cached_artifacts(self):
        import os
        for f in ["model.pkl", "imputer.pkl",
"feature_columns.json", "tipos_maestro.json"]:
            if os.path.exists(f):
                try:
                    os.remove(f)
                    print(f"[INFO] Eliminado: {f}")
                except Exception as e:
                    print(f"[WARN] No pude eliminar {f}: {e}")

    def load_data(self, sheet_name: str | None = None, reset: bool
= True):
        """Carga los datos desde el Excel y opcionalmente reinicia
el estado previo."""

        import pandas as pd
        import numpy as np
        import os

        if reset:
            self.reset_state()
            self.delete_cached_artifacts()

        if hasattr(self, "data_path"):
            path = self.data_path
        elif hasattr(self, "input_file"):
            path = self.input_file
        else:
            raise ValueError("No existe data_path ni input_file en
este objeto.")

        if path is None:
            raise ValueError("La ruta del archivo de datos es
None.")
        if not os.path.isfile(path):
            raise FileNotFoundError(f"No se encontró el archivo de
datos: {path}")

        xls = pd.ExcelFile(path)

        if sheet_name:
            df = pd.read_excel(xls, sheet_name=sheet_name)
        else:
```

```
            df = pd.read_excel(xls)

        df = df.replace({pd.NA: np.nan}).infer_objects(copy=False)
        df = df.dropna(axis=1, how="all")


        self.df_raw = df
        self.df_original = df.copy()
        self.input_file = path
        self.data_path = path

        print(f"[INFO] Archivo cargado correctamente: {path}
(filas: {len(df)})")
        return df

    def preprocess_data(self, learn_types=False):
        assert self.df_original is not None, "Primero ejecuta
load_data()"
        self.df_processed = self._process_df(self.df_original,
learn_types=True)
        if "Recuperación Cu (%)" in self.df_processed.columns:
            self.feature_columns = [c for c in
self.df_processed.columns if c != "Recuperación Cu (%)"]
        else:
            self.feature_columns = list(self.df_processed.columns)

    def train_model(self):
        assert self.df_processed is not None, "Primero ejecuta
preprocess_data()"
        if "Recuperación Cu (%)" not in self.df_processed.columns:
            raise ValueError("Falta 'Recuperación Cu (%)' en los
datos para entrenar")

        df_model = self.df_processed.dropna(subset=["Recuperación
Cu (%)"])
        if df_model.shape[0] < 10:
            raise ValueError("No hay suficientes filas con
'Recuperación Cu (%)' para entrenar (mín=10)")

        y = self.df_processed["Recuperación Cu (%)"].copy()
        y = pd.to_numeric(y, errors="coerce")

        y_max = y.dropna().max()
        if pd.notna(y_max) and y_max <= 1.0:
            y = y * 100
```

```
        X = self.df_processed[self.feature_columns].copy()
        X = X.replace({pd.NA: np.nan})
        X = X.apply(pd.to_numeric, errors="coerce")

        X = X.loc[:, X.notna().any(axis=0)]

        mask = y.notna()
        X = X.loc[mask]
        y = y.loc[mask]

        X = X.reset_index(drop=True)
        y = y.reset_index(drop=True)

        self.model_features = X.columns.tolist()

        self.imputer = SimpleImputer(strategy="mean")
        X_imp = self.imputer.fit_transform(X[self.model_features])

        X_train, X_test, y_train, y_test = train_test_split(
            X_imp, y, test_size=0.2, random_state=42
        )

        self.model = RandomForestRegressor(n_estimators=200,
random_state=42)
        self.model.fit(X_train, y_train)

        y_pred = self.model.predict(X_test)

    def predict_from_dict(self, sample_dict: dict) -> float:
        """Predice recuperación para un caso nuevo usando el MISMO
esquema de columnas que en entrenamiento."""
        assert (self.model is not None) and (self.imputer is not
None) and (self.feature_columns is not None) and
(self.tipos_maestro is not None), \
            "Modelo/imputador no entrenados o tipos no fijados.
Ejecuta load_data() → preprocess_data() → train_model()."

        if not hasattr(self, "model_features") or
self.model_features is None:
            raise RuntimeError("El modelo no ha sido entrenado:
self.model_features no está definido.")

        df_new = pd.DataFrame([sample_dict])
        df_proc = self._process_df(df_new, learn_types=False)

        df_proc = df_proc.replace({pd.NA: np.nan})
```

```
        df_proc = df_proc.apply(pd.to_numeric, errors="coerce")

        df_proc = df_proc.reindex(columns=self.model_features,
fill_value=np.nan)

        X_new = self.imputer.transform(df_proc)
        y_hat = float(self.model.predict(X_new)[0])

        if y_hat <= 1.0:
            y_hat *= 100

        return round(y_hat, 2)

    def predict_interactive(self):
        """Predice en bucle. Feedback: cada test se agrega y el
modelo se reentrena al instante (REAL si lo provees; si no,
PRED)."""
        os.makedirs(self.outdir, exist_ok=True)

        session_preds = []
        session_truths = []

        def ask(prompt):
            try:
                return input(prompt).strip()
            except EOFError:
                return ""

        def none_if_empty(s):
            return None if s == "" else s

        while True:
            print("\nIngrese datos del nuevo test (ENTER para
omitir). Use comas para múltiples valores.")
            sample = {}
            sample["pH"] = none_if_empty(ask("pH: "))

            sample["Velocidad gas"] =
none_if_empty(ask("Velocidad/caudal de gas (numérico): "))
            sample["Unidad gas"] = none_if_empty(ask("Unidad gas
(m/s, cm/s, m/min, m3/h, L/min...). Si vacío, se asume m/s: "))

            sample["Tiempo de flotación"] =
none_if_empty(ask("Tiempo de flotación (min): "))
            sample["Tiempo de acondicionamiento"] =
none_if_empty(ask("Tiempo de acondicionamiento (min): "))
```

```
            sample["Tamaño de partícula"] =
none_if_empty(ask("Tamaño de partícula (ej: 150 | 0.1-0.3 | d80 |
<200): "))
            sample["Unidad tam"] = none_if_empty(ask("Unidad
tamaño (mm o µm): "))

            def handle_group(nombre, col_tipo, col_dosis,
col_unidad, gkey):
                tipos_sel =
self._interactive_select_options(nombre + " (tipos)",
sorted(self.tipos_maestro.get(gkey, set())), allow_multi=True,
group_key=gkey)
                if len(tipos_sel) == 0:
                    return
                sample[col_tipo] = ", ".join(tipos_sel)

                mezcla = ask(f"¿Usar una sola dosis para la mezcla
de {nombre}? [s/N]: ").lower() == 's'
                if mezcla:
                    dose = none_if_empty(ask("Dosis única para la
mezcla (numérico): "))
                    unit = self._interactive_select_unit(f"Unidad
para la dosis de {nombre}")
                    sample[col_dosis] = dose
                    sample[col_unidad] = unit
                    return

                dosis_list = []
                for t in tipos_sel:
                    d = none_if_empty(ask(f"Dosis para {t}
(numérico): "))
                    dosis_list.append(d if d is not None else "")
                sample[col_dosis] = ", ".join([x for x in
dosis_list if x is not None])

                una_unidad = ask("¿Una sola unidad para todos los
tipos? [S/n]: ").lower() != 'n'
                if una_unidad:
                    unit = self._interactive_select_unit(f"Unidad
(aplicará a todos los {nombre})")
                    sample[col_unidad] = unit
                else:
                    unidades_list = []
                    for t in tipos_sel:
```

```
                        u = self._interactive_select_unit(f"Unidad
para {t}")
                        unidades_list.append(u)
                    sample[col_unidad] = ", ".join(unidades_list)

            handle_group("Colector", "Tipo de colector", "Dosis
colector", "Unidad col", "colector")
            handle_group("Depresante", "Tipo de depresante",
"Dosis depresante", "Unidad dep", "depresante")
            handle_group("Espumante", "Tipo de espumante", "Dosis
espumante", "Unidad esp", "espumante")
            handle_group("Otros reactivos", "Otros reactivos",
"Dosis otros reactivos", "Unidad reac", "otros")

            def handle_simple_cat(nombre, col, gkey):
                sels = self._interactive_select_options(nombre,
sorted(self.tipos_maestro.get(gkey, set())), allow_multi=True,
group_key=gkey)
                if len(sels) > 0:
                    sample[col] = ", ".join(sels)
            handle_simple_cat("Tipo de agua", "Tipo de agua",
"agua")
            handle_simple_cat("Tipo de mineral", "Tipo de
mineral", "mineral")
            handle_simple_cat("Tipo de celda", "Tipo de celda",
"celda")

            for k in ["pH", "Velocidad gas", "Tiempo de
flotación", "Tiempo de acondicionamiento"]:
                if sample.get(k) not in (None, ""):
                    try:
                        sample[k] =
float(str(sample[k]).replace(",", "."))
                    except Exception:
                        pass

            y_hat = self.predict_from_dict(sample)
            print(f"\n>>> Recuperación Cu (%) estimada:
{y_hat:.2f}\n")

            tiene_real = ask("¿Tienes el valor REAL de
recuperación para este test? [s/N]: ").lower() == 's'
            if tiene_real:
                real_str = ask("Ingresa la recuperación real (%):
")
                try:
```

```
                    real_val = float(str(real_str).replace(",",
"."))
                except Exception:
                    real_val = np.nan
            else:
                real_val = np.nan

            session_preds.append(float(y_hat))
            session_truths.append(real_val)

            y_to_use = float(real_val) if not np.isnan(real_val)
else float(y_hat)
            source = "feedback_real" if not np.isnan(real_val)
else "feedback_pred"
            self._append_feedback_and_retrain(sample, y_to_use,
source)

            continuar = ask("¿Quieres predecir OTRO test? [s/N]:
").lower() == 's'
            if not continuar:
                break

        self._plot_session_results(session_preds, session_truths)
        self._save_user_vocab()

    def _plot_session_results(self, preds, truths):
        """Genera PNG y XLSX de comparación Predicho vs Real si
hubo al menos un REAL."""
        arr_p = np.array(preds, dtype=float)
        arr_t = np.array(truths, dtype=float)
        mask = ~np.isnan(arr_t)
        if mask.sum() == 0:
            print("[INFO] No se ingresaron valores reales; no se
genera comparativa.")
            return

        idx = np.arange(1, len(arr_p) + 1)[mask]
        p_ok = arr_p[mask]
        t_ok = arr_t[mask]

        ts = datetime.now().strftime("%Y%m%d_%H%M%S")
        png_path = os.path.join(self.outdir,
f"comparativa_pred_vs_real_{ts}.png")
        plt.figure(figsize=(10, 6))
        plt.plot(idx, t_ok, marker='o', label='Real')
        plt.plot(idx, p_ok, marker='x', label='Predicho')
```

```
        plt.xlabel("Test # (sesión)")
        plt.ylabel("Recuperación Cu (%)")
        plt.title(f"Predicho vs Real")
        plt.legend()
        plt.tight_layout()
        plt.savefig(png_path, dpi=150)
        plt.close()
        print(f"[OK] PNG guardado: {png_path}")

        xlsx_path = os.path.join(self.outdir,
f"comparativa_pred_vs_real_{ts}.xlsx")
        df_cmp = pd.DataFrame({"Test_sesion": idx, "Real": t_ok,
"Predicho": p_ok})
        with pd.ExcelWriter(xlsx_path, engine="openpyxl") as
writer:
            df_cmp.to_excel(writer, index=False,
sheet_name="comparativa")
            df_sum.to_excel(writer, index=False,
sheet_name="resumen")
        print(f"[OK] XLSX guardado: {xlsx_path}")

    def _append_feedback_and_retrain(self, sample_original_format:
dict, y_value: float, source: str):
        """Agrega una fila a la base aumentada (Excel) y reentrena
el modelo."""
        row = dict(sample_original_format)
        row["Recuperación Cu (%)"] = y_value
        row["_source"] = source
        row["_timestamp"] =
datetime.now().isoformat(timespec="seconds")

        os.makedirs(self.outdir, exist_ok=True)
        if os.path.exists(self.augmented_path):
            try:
                df_fb = pd.read_excel(self.augmented_path,
sheet_name="data")
            except Exception:
                df_fb = pd.DataFrame()
            df_fb = pd.concat([df_fb, pd.DataFrame([row])],
ignore_index=True, sort=False)
        else:
            df_fb = pd.DataFrame([row])

        try:
            with pd.ExcelWriter(self.augmented_path,
engine="openpyxl") as writer:
```

```
                df_fb.to_excel(writer, sheet_name="data",
index=False)
            print(f"[INFO] Feedback añadido a
{self.augmented_path} (total filas: {len(df_fb)})")
        except Exception as e:
            print(f"[WARN] No se pudo escribir el feedback
aumentado ({e}).")

        if self.df_original is None:
            self.df_original = pd.DataFrame([row])
        else:
            self.df_original = pd.concat([self.df_original,
pd.DataFrame([row])], ignore_index=True, sort=False)

        self.df_processed = self._process_df(self.df_original,
learn_types=True)
        if "Recuperación Cu (%)" in self.df_processed.columns:
            self.feature_columns = [c for c in
self.df_processed.columns if c != "Recuperación Cu (%)"]
        else:
            self.feature_columns = list(self.df_processed.columns)

        try:
            self.train_model()
            print("[INFO] Modelo reentrenado con base aumentada.")
        except Exception as e:
            print(f"[WARN] No fue posible reentrenar
inmediatamente ({e}). El feedback queda guardado.")

    def _fresh_model(self):
        self.model = RandomForestRegressor(n_estimators=200,
random_state=42)
        self.imputer = SimpleImputer(strategy="mean")

    def _XY(self):
        assert self.df_processed is not None and
self.feature_columns is not None
        if "Recuperación Cu (%)" not in self.df_processed.columns:
            raise ValueError("La base no tiene 'Recuperación Cu
(%)' para evaluar")
        dfm = self.df_processed.dropna(subset=["Recuperación Cu
(%)"]).copy()
        X = dfm[self.feature_columns].copy()
        y = dfm["Recuperación Cu (%)"].astype(float).values
        return X, y
```

```
float(self.model.predict(self.imputer.transform(Xi))[0])
            preds.append((yi, yhat))
            if verbose:
                print(f"{k:3d}/{len(test_idx)}  y_real={yi:.2f}  y
_pred={yhat:.2f}")
            train_idx = np.append(train_idx, i)
            self._fresh_model()
            X_train = X.iloc[train_idx]
            self.imputer.fit(X_train)
            Xti = self.imputer.transform(X_train)
            self.model.fit(Xti, y[train_idx])

        y_true = np.array([a for a, _ in preds])
        y_pred = np.array([b for _, b in preds])

    def _fit_eval_split(self, X, y, train_idx, test_idx,
verbose=True):
        self._fresh_model()
        self.imputer.fit(X.iloc[train_idx])
        Xtr = self.imputer.transform(X.iloc[train_idx])
        Xte = self.imputer.transform(X.iloc[test_idx])
        self.model.fit(Xtr, y[train_idx])
        yhat = self.model.predict(Xte)

    def _norm_unit(self, u: str):
        if pd.isna(u):
            return None
        u = str(u).strip().lower().replace(" ", "")
        aliases = {
            "g/t": "g/t", "gr/t": "g/t", "gpt": "g/t", "gport":
"g/t",
            "kg/t": "kg/t", "kgt": "kg/t",
            "mg/t": "mg/t",
            "mg/l": "mg/l", "g/l": "g/l",
            "kg/m3": "kg/m3", "kg/m^3": "kg/m3"
        }
        return aliases.get(u, u)

    def _unit_family_and_factor(self, unit_norm: str):
        if unit_norm in ("mg/t", "g/t", "kg/t"):
            return "mass_ton", {"mg/t": 0.001, "g/t": 1.0, "kg/t":
1000.0}[unit_norm], "g/t"
        if unit_norm in ("mg/l", "g/l", "kg/m3"):
            return "mass_vol", {"mg/l": 1.0, "g/l": 1000.0,
"kg/m3": 1000.0}[unit_norm], "mg/l"
        return "unknown", None, None
```

```
    def _norm_unit_gas(self, u: str | None):
        if pd.isna(u) or u == "":
            return None
        s = str(u).strip().lower().replace(" ", "")
        s = s.replace("m³", "m3").replace("^3", "3")
        s = s.replace("ℓ", "l")
        return s

    def _convert_gas(self, val, unit_raw):
        """
        Devuelve: (vel_mps, flow_m3s, is_flow, mismatch,
assumed_mps)
        - vel_mps: velocidad en m/s o NaN
        - flow_m3s: caudal en m3/s o NaN
        - is_flow: True si la unidad era de caudal (no velocidad)
        - mismatch: True si la unidad es desconocida/ambigua
        - assumed_mps: True si no hubo unidad y asumimos m/s
        """
        try:
            x = float(str(val).replace(",", ".")) if not
pd.isna(val) and val != "" else None
        except Exception:
            x = None

        u = self._norm_unit_gas(unit_raw)
        if x is None:
            return (pd.NA, pd.NA, False, False, False)

        vel_map = {
            "m/s": 1.0,
            "cm/s": 0.01,
            "mm/s": 0.001,
            "m/min": 1.0 / 60.0,
            "cm/min": 0.01 / 60.0,
            "mm/min": 0.001 / 60.0,
            "m/h": 1.0 / 3600.0,
            "cm/h": 0.01 / 3600.0,
            "mm/h": 0.001 / 3600.0,
        }

        flow_map = {
            "m3/s": 1.0,
            "m3/h": 1.0 / 3600.0,
            "l/s": 1e-3,
            "l/min": 1e-3 / 60.0,
```

```
            "l/h": 1e-3 / 3600.0,
        }

        if u in vel_map:
            return (x * vel_map[u], pd.NA, False, False, False)
        if u in flow_map:
            return (pd.NA, x * flow_map[u], True, False, False)

        if u is None:
            return (x, pd.NA, False, False, True)

        return (pd.NA, pd.NA, False, True, False)

    def _parse_size_value(self, val):
        if pd.isna(val):
            return pd.NA
        s = str(val).strip().lower().replace(" ", "")
        try:
            if s.startswith("<"):
                return float(s[1:])
            if "-" in s:
                a, b = s.split("-", 1)
                return (float(a) + float(b)) / 2.0
            # d80, p80 → tomar número
            if s.startswith("d") or s.startswith("p"):
                return float(s[1:])
            return float(s)
        except Exception:
            return pd.NA

    def _to_um(self, value, unit_str):
        if pd.isna(value):
            return pd.NA
        unit = str(unit_str).strip().lower() if not
pd.isna(unit_str) else ""
        if "mm" in unit:
            return float(value) * 1000.0
        return float(value)  # default µm

    def _process_df(self, df_in: pd.DataFrame, learn_types: bool)
-> pd.DataFrame:
        df = df_in.copy()

        def _safe_col(frame, name, fill=""):
            """Devuelve una Series del frame si existe; si no, una
Series rellena con `fill` y mismo índice."""
```

```
            if name in frame.columns:
                return frame[name].fillna(fill)
            else:
                return pd.Series(fill, index=frame.index)

        cols_meta = [
            "Autor", "Año", "Ano", "País", "Pais", "Enlace",
"URL", "Link",
            "Fuente", "Referencia", "Paper", "Referencia URL"
        ]
        df.drop(
            columns=[c for c in cols_meta if c in df.columns],
            inplace=True,
            errors="ignore"
        )


        if "Tamaño de partícula" in df.columns:
            df["Tam_prom_raw"] = df["Tamaño de
partícula"].apply(self._parse_size_value)
        if "Unidad tam" in df.columns:
            df["Tam_um"] = df.apply(lambda r:
self._to_um(r.get("Tam_prom_raw"), r.get("Unidad tam")), axis=1)
        df["Particula_FinaGruesa"] = df["Tam_um"].apply(lambda x:
1 if (pd.notna(x) and float(x) > 200.0) else (0 if pd.notna(x)
else pd.NA))

        if "Velocidad gas" in df.columns:
            df["Velocidad_gas_mps"] = pd.NA
            df["Caudal_gas_m3s"] = pd.NA
            df["vel_gas_flag_is_flow"] = 0
            df["vel_gas_flag_unit_mismatch"] = 0
            df["vel_gas_flag_assumed_mps"] = 0

            def _row_gas(r):
                vel, flow, is_flow, mis, assumed =
self._convert_gas(r.get("Velocidad gas"), r.get("Unidad gas"))
                r["Velocidad_gas_mps"] = vel
                r["Caudal_gas_m3s"] = flow
                r["vel_gas_flag_is_flow"] = 1 if is_flow else 0
                r["vel_gas_flag_unit_mismatch"] = 1 if mis else 0
                r["vel_gas_flag_assumed_mps"] = 1 if assumed else
0
                return r

            df = df.apply(_row_gas, axis=1)
```

```
        cat_cols = [("Tipo de agua", "agua__"), ("Tipo de
mineral", "mineral__"), ("Tipo de celda", "celda__")]
        if learn_types:
            self.tipos_maestro = {"colector": set(), "depresante":
set(), "espumante": set(), "otros": set(),
                                  "agua": set(), "mineral": set(),
"celda": set()}
        assert self.tipos_maestro is not None

        for col, prefix in cat_cols:
            if learn_types and col in df.columns:
                for lista in
df[col].dropna().astype(str).str.split(','):
                    self.tipos_maestro[prefix[:-
2]].update([x.strip() for x in lista if x.strip() != ""])

        for k in self.tipos_maestro.keys():
            self.tipos_maestro[k].update(self._user_vocab.get(k,
set()))

        for col, prefix in cat_cols:
            tipos = sorted(self.tipos_maestro.get(prefix[:-2],
[])) if self.tipos_maestro else []
            col_series = _safe_col(df, col, "")
            split_series = col_series.astype(str).apply(lambda s:
[x.strip() for x in s.split(',') if x.strip() != ""])
            for t in tipos:
                df[f"{prefix}{t}"] = split_series.apply(lambda
lst: 1 if t in lst else 0)

        grupos = [
            ("Tipo de colector", "Dosis colector", "Unidad col",
"colector"),
            ("Tipo de depresante", "Dosis depresante", "Unidad
dep", "depresante"),
            ("Tipo de espumante", "Dosis espumante", "Unidad esp",
"espumante"),
            ("Otros reactivos", "Dosis otros reactivos", "Unidad
reac", "otros")
        ]

        for col_tipo, _, _, g in grupos:
            if learn_types and col_tipo in df.columns:
                for lista in
df[col_tipo].dropna().astype(str).str.split(','):
```

```
                    self.tipos_maestro[g].update([x.strip() for x
in lista if x.strip() != ""])
        for g in ("colector", "depresante", "espumante", "otros"):
            self.tipos_maestro[g].update(self._user_vocab.get(g,
set()))

        for col_tipo, col_dosis, col_unidad, g in grupos:
            tipos = sorted(self.tipos_maestro[g])

            col_series_tipo = _safe_col(df, col_tipo, "")
            split_series_tipo = col_series_tipo.astype(str).apply(
                lambda s: [x.strip() for x in s.split(',') if
x.strip() != ""]
            )

            batch_cols = {}
            for t in tipos:
                batch_cols[f"{g}__{t}"] =
split_series_tipo.apply(lambda lst: 1 if t in lst else 0)

            if batch_cols:
                df = pd.concat([df, pd.DataFrame(batch_cols,
index=df.index)], axis=1)

            flag_cols = {
                f"{g}_dose_total_gpt": pd.Series(pd.NA,
index=df.index, dtype="Float64"),
                f"{g}_dose_total_mgl": pd.Series(pd.NA,
index=df.index, dtype="Float64"),
                f"{g}_flag_mezcla":     pd.Series(0,     index=df.
index, dtype="Int64"),
                f"{g}_flag_mismatch":   pd.Series(0,     index=df.
index, dtype="Int64"),
                f"{g}_flag_unit_mismatch": pd.Series(0,
index=df.index, dtype="Int64"),
                f"{g}_flag_sin_tipo":   pd.Series(0,     index=df.
index, dtype="Int64"),
            }
            df = pd.concat([df, pd.DataFrame(flag_cols,
index=df.index)], axis=1)
            df = df.copy()

            created = set(batch_cols.keys()) |
set(flag_cols.keys())
```

```
            def fila_proc(row):
                tipos_l = [x.strip() for x in
str(row.get(col_tipo, "")).split(',') if x.strip() != ""] \
                    if not pd.isna(row.get(col_tipo)) else []
                dosis_l = [x.strip() for x in
str(row.get(col_dosis, "")).split(',') if x.strip() != ""] \
                    if not pd.isna(row.get(col_dosis)) else []
                unidades_l = [self._norm_unit(u) for u in
                              ([x.strip() for x in
str(row.get(col_unidad, "")).split(',') if x.strip() != ""]
                               if not pd.isna(row.get(col_unidad))
else [])]

                if len(tipos_l) == 0:
                    if row.get(col_dosis) not in (None, "", "-")
and not pd.isna(row.get(col_dosis)):
                        row[f"{g}_flag_sin_tipo"] = 1
                    return row

                if len(unidades_l) == 1 and len(tipos_l) > 1:
                    unidades_l = unidades_l * len(tipos_l)
                if len(dosis_l) == 1 and len(tipos_l) > 1:
                    row[f"{g}_flag_mezcla"] = 1

                if not ((len(dosis_l) in (0, 1, len(tipos_l))) and
(len(unidades_l) in (0, 1, len(tipos_l)))):
                    row[f"{g}_flag_mismatch"] = 1

                total_gpt = 0.0; any_gpt = False
                total_mgl = 0.0; any_mgl = False

                for idx, t in enumerate(tipos_l):
                    # dosis
                    if len(dosis_l) == len(tipos_l):
                        d_raw = dosis_l[idx]
                    elif len(dosis_l) == 1 and len(tipos_l) > 1:
                        d_raw = None
                    else:
                        d_raw = None
                    try:
                        d_num = float(str(d_raw).replace(",",
".")) if d_raw not in (None, "", "-") else None
                    except Exception:
                        d_num = None

                    # unidad
```

```
                    if len(unidades_l) == len(tipos_l):
                        u = unidades_l[idx]
                    elif len(unidades_l) == 1 and len(tipos_l) >=
1:
                        u = unidades_l[0]
                    else:
                        u = None

                    if u is not None:
                        fam, factor, _ =
self._unit_family_and_factor(u)
                    else:
                        fam, factor, _ = ("unknown", None, None)

                    col_gpt = f"{g}_dose_gpt__{t}"
                    col_mgl = f"{g}_dose_mgl__{t}"
                    if col_gpt not in created:
                        df[col_gpt] = pd.NA; created.add(col_gpt)
                    if col_mgl not in created:
                        df[col_mgl] = pd.NA; created.add(col_mgl)

                    if d_num is not None and fam == "mass_ton":
                        row[col_gpt] = d_num * factor
                        total_gpt += float(row[col_gpt]); any_gpt
= True
                    elif d_num is not None and fam == "mass_vol":
                        row[col_mgl] = d_num * factor
                        total_mgl += float(row[col_mgl]); any_mgl
= True
                    else:
                        if d_num is not None:
                            row[f"{g}_flag_unit_mismatch"] = 1

                row[f"{g}_dose_total_gpt"] = total_gpt if any_gpt
else pd.NA
                row[f"{g}_dose_total_mgl"] = total_mgl if any_mgl
else pd.NA
                return row

            df = df.apply(fila_proc, axis=1)

        # Convertir numéricos típicos
        for col in ["pH", "Tiempo de flotación", "Tiempo de
acondicionamiento", "Tam_um", "Particula_FinaGruesa",
                    "Velocidad_gas_mps", "Caudal_gas_m3s",
"vel_gas_flag_is_flow",
```

```
                    "vel_gas_flag_unit_mismatch",
"vel_gas_flag_assumed_mps"]:
            if col in df.columns:
                df[col] = pd.to_numeric(df[col], errors='coerce')

        # Quitar columnas originales ya usadas/irrelevantes
        df.drop(columns=[
            "Tamaño de partícula", "Unidad tam",
            "Tipo de colector", "Dosis colector", "Unidad col",
            "Tipo de depresante", "Dosis depresante", "Unidad
dep",
            "Tipo de espumante", "Dosis espumante", "Unidad esp",
            "Otros reactivos", "Dosis otros reactivos", "Unidad
reac",
            "Tipo de agua", "Tipo de mineral", "Tipo de celda",
            "Velocidad gas", "Unidad gas",
            "Unidad flot", "Unidad ac"
        ], errors='ignore', inplace=True)

        # Fijar catálogo aprendido
        if learn_types and self.tipos_maestro is None:
            self.tipos_maestro = {"colector": set(), "depresante":
set(), "espumante": set(), "otros": set(),
                                  "agua": set(), "mineral": set(),
"celda": set()}
        df = df.replace({pd.NA: np.nan}).infer_objects(copy=False)
        df = df.dropna(axis=1, how="all")
        return df

    # --------- selección interactiva ---------
    def _interactive_select_options(self, titulo: str, opciones:
list[str], allow_multi: bool = True, group_key: str | None = None)
-> list[str]:
        print(f"\n— {titulo} —")
        if not opciones:
            print("(No hay opciones en la base aún)")
        else:
            for i, opt in enumerate(opciones, start=1):
                print(f"  {i}) {opt}")
        print("  0) otro (agregar nuevo)")
        s = input("Elige número(s) separados por coma o ENTER para
saltar: ").strip()
        if s == "":
            return []
        idxs = [x.strip() for x in s.split(',') if x.strip() !=
""]
```

```
        seleccion = []
        for idx in idxs:
            if idx == '0':
                nuevo = input("Escribe nuevo(s) nombre(s)
separados por coma: ").strip()
                nuevos = [x.strip() for x in nuevo.split(',') if
x.strip() != ""]
                seleccion.extend(nuevos)
                if group_key:
                    self._user_vocab.setdefault(group_key,
set()).update(nuevos)
                    if self.tipos_maestro and group_key in
self.tipos_maestro:
                        self.tipos_maestro[group_key].update(nuevo
s)
            else:
                try:
                    i = int(idx)
                    if 1 <= i <= len(opciones):
                        seleccion.append(opciones[i - 1])
                except Exception:
                    pass
        # quitar duplicados preservando orden
        seen = set(); out = []
        for x in seleccion:
            if x not in seen:
                out.append(x); seen.add(x)
        return out

    def _interactive_select_unit(self, titulo: str) -> str:
        units = ["mg/t", "g/t", "kg/t", "mg/L", "g/L", "kg/m3"]
        print(f"\n— {titulo} —")
        for i, u in enumerate(units, start=1):
            print(f"  {i}) {u}")
        print("  0) otro")
        s = input("Elige 1 opción (número) o escribe directamente
la unidad: ").strip()
        if s == "":
            return ""
        if s.isdigit():
            i = int(s)
            if i == 0:
                u = input("Escribe la unidad (ej: g/t): ").strip()
                return u
            if 1 <= i <= len(units):
                return units[i - 1]
```

```
        return s

    def _load_user_vocab(self):
        try:
            if os.path.exists(self.user_vocab_path):
                with open(self.user_vocab_path, 'r',
encoding='utf-8') as f:
                    data = json.load(f)
                return {k: set(v) for k, v in data.items()}
        except Exception:
            pass
        return {"colector": set(), "depresante": set(),
"espumante": set(), "otros": set(),
                "agua": set(), "mineral": set(), "celda": set()}

    def _save_user_vocab(self):
        try:
            data = {k: sorted(list(v)) for k, v in
self._user_vocab.items()}
            with open(self.user_vocab_path, 'w', encoding='utf-8')
as f:
                json.dump(data, f, ensure_ascii=False, indent=2)
        except Exception:
            pass


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="IA de flotación:
entrenamiento, predicción y evaluación. v4.3")

    parser.add_argument("--data", required=False,
default="datos_flotacion_completo2.xlsx", help="Ruta al Excel de
datos")
    parser.add_argument("--sheet", default=None, help="Nombre de
hoja (opcional)")
    parser.add_argument("--train_frac", type=float, default=0.7,
help="Fracción de entrenamiento para los experimentos")
    parser.add_argument("--seed", type=int, default=42,
help="Semilla aleatoria")
    parser.add_argument("--load_model", default=None, help="Ruta a
modelo .pkl ya entrenado (opcional)")
    parser.add_argument("--load_imputer", default=None, help="Ruta
a imputador .pkl ya entrenado (opcional)")
    parser.add_argument("--outdir", default="./outputs",
help="Carpeta para guardar PNG/XLSX y feedback")
```

```
    parser.add_argument("--augmented_path", default=None,
help="Ruta personalizada para el Excel de feedback aumentado
(opcional)")
    parser.add_argument("--verbose", action="store_true",
help="Más detalle en 'incremental'")

    parser.add_argument("--fresh", action="store_true",
        help="No cargar feedback ni vocabulario previos (arranque
limpio en esta ejecución; usa archivos fresh en outdir)")
    parser.add_argument("--wipe", action="store_true",
        help="Eliminar feedback y vocabulario guardados antes de
arrancar (y luego correr en limpio)")

    args = parser.parse_args()

    os.makedirs(args.outdir, exist_ok=True)

    if args.wipe:
        aug_path_default = args.augmented_path or
os.path.join(args.outdir, "augmented_feedback.xlsx")
        if os.path.exists(aug_path_default):
            try:
                os.remove(aug_path_default)
                print(f"[INFO] Eliminado {aug_path_default}")
            except Exception as e:
                print(f"[WARN] No se pudo eliminar
{aug_path_default}: {e}")
        if os.path.exists("./user_vocab.json"):
            try:
                os.remove("./user_vocab.json")
                print("[INFO] Eliminado ./user_vocab.json")
            except Exception as e:
                print(f"[WARN] No se pudo eliminar
./user_vocab.json: {e}")

    user_vocab_path = "./user_vocab.json"
    augmented_path = args.augmented_path
    if args.fresh:
        user_vocab_path = os.path.join(args.outdir,
"user_vocab_fresh.json")
        augmented_path = os.path.join(args.outdir,
"augmented_feedback_fresh.xlsx")

# Inicializar el código
ai = FlotationAI(
    args.data,
```

```
    user_vocab_path=user_vocab_path,
    outdir=args.outdir,
    augmented_path=augmented_path
)

# Cargar datos
print("[INFO] Ejecutando load_data()...")
ai.load_data(sheet_name=args.sheet, reset=True)

if ai.df_original is None:
    raise RuntimeError("[ERROR]")

print("[INFO] Ejecutando preprocess_data()...")
ai.preprocess_data()
```

Anexo B: Parámetros de flotación utilizados como database.

Test 1:
- Recuperación: 88,58%.
- pH: 7.
- Tipo de colector: SIBX.
- Dosis de colector: 10 g/t.
- Tipo de depresante: Na_2SiO_3.
- Dosis de depresante: 50 g/t.
- Tipo de espumante: Hydrofroth.
- Dosis de espumante: 10 g/t.
- Otros reactivos: No entregado.
- Dosis de otros reactivos: No entregado.
- Tipo de agua: Agua.
- Tipo de mineral: Relaves.
- Tipo de celda: Mecánica.
- Tamaño de partícula: Menor a 75 µm.
- Velocidad gas: 7 L/min.
- Tiempo de flotación: 2 min.
- Tiempo de acondicionamiento: 7 min.

Test 2:
- Recuperación: 47,02%.
- pH: 7.
- Tipo de colector: SIBX.
- Dosis de colector: 100 g/t.
- Tipo de depresante: Na_2SiO_3.
- Dosis de depresante: 150 g/t.
- Tipo de espumante: Hydrofroth.
- Dosis de espumante: 10 g/t.
- Otros reactivos: No entregado.
- Dosis de otros reactivos: No entregado.
- Tipo de agua: Agua.
- Tipo de mineral: Relaves.
- Tipo de celda: Mecánica.
- Tamaño de partícula: Menor a 75 µm.
- Velocidad gas: 7 L/min.
- Tiempo de flotación: 2 min.
- Tiempo de acondicionamiento: 7 min.

Test 3:
- Recuperación: 60,00%.
- pH: 7.
- Tipo de colector: SIBX.
- Dosis de colector: 80 g/t.
- Tipo de depresante: Na2SiO3.
- Dosis de depresante: 150 g/t.
- Tipo de espumante: Hydrofroth.
- Dosis de espumante: 30 g/t.

Test 4:
- Recuperación: 63,20%.
- pH: 11.
- Tipo de colector: SIPX, DBS.
- Dosis de colector: No entregado.
- Tipo de depresante: No entregado.
- Dosis de depresante: No entregado.
- Tipo de espumante: MIBC, Nasfroth.
- Dosis de espumante: No entregado.

• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Relaves.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 75 µm.
• Velocidad gas: 7 L/min
• Tiempo de flotación: 2 min.
• Tiempo de acondicionamiento: 7 min.

Test 5:
• Recuperación: 62,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 7:
• Recuperación: 62,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 6 kg/t
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 6:
• Recuperación: 63,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 8:
• Recuperación: 63,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 9:
• Recuperación: 62,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 10:
• Recuperación: 52,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 11:
• Recuperación: 50,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 12:
• Recuperación: 53,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1.5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 13:
• Recuperación: 65,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.

Test 14:
• Recuperación: 54,20%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.

• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 15:
• Recuperación: 63,20%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 17:
• Recuperación: 62,60%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.

• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 16:
• Recuperación: 64,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 18:
• Recuperación: 67,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.

• Tiempo de acondicionamiento: 3 min.

Test 19:
• Recuperación: 64,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 20:
• Recuperación: 54,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: SiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 21:
• Recuperación: 68,20%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 22:
• Recuperación: 60,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 23:
• Recuperación: 53,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.

Test 24:
• Recuperación: 65,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.

• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 25:
• Recuperación: 64,50%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 26:
• Recuperación: 64,80%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3.0

Test 27:
• Recuperación: 51,20%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.

Test 28:
• Recuperación: 51,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.

• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 29:
• Recuperación: 52,90%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 30:
• Recuperación: 55,10%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 31:
• Recuperación: 54,00%
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 32:
• Recuperación: 52,80%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 33:
• Recuperación: 64,90%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.

Test 34:
• Recuperación: 65,20%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.

• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 35:
• Recuperación: 55,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 37:
• Recuperación: 53,70%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.

• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 36:
• Recuperación: 54,10%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 38:
• Recuperación: 59,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.

• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 39:
• Recuperación: 55,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 41:
• Recuperación: 65,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 43:
• Recuperación: 64,50%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 40:
• Recuperación: 60,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3,
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 42:
• Recuperación: 69,20%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 44:
• Recuperación: 60,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.

• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 45:
• Recuperación: 54,50%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 47:
• Recuperación: 64,00%
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.

• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 46:
• Recuperación: 55,60%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 48:
• Recuperación: 60,30%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.

• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 49:
• Recuperación: 61,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 51:
• Recuperación: 58,10%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 53:
• Recuperación: 62,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.

• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 50:
• Recuperación: 55,80%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 52:
• Recuperación: 70,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 54:
• Recuperación: 57,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.

• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 55:
• Recuperación: 64,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 57:
• Recuperación: 63,00%
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.

• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 56:
• Recuperación: 62,50%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 58:
• Recuperación: 62,70%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.

• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 59:
• Recuperación: 63,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 61:
• Recuperación: 60,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 63:
• Recuperación: 51,50%.
• pH: 10.

• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 60:
• Recuperación: 62,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 62:
• Recuperación: 58,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 64:
• Recuperación: 72,00%.
• pH: 12.

• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 65:
• Recuperación: 54,20%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 67:
• Recuperación: 64,00%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.

• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 66:
• Recuperación: 63,20%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 68:
• Recuperación: 62,60%.
• pH: 11.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.

• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 69:
• Recuperación: 70,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 71:
• Recuperación: 54,00%.
• pH: 10.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 70:
• Recuperación: 67,00%.
• pH: 12.
• Tipo de colector: SIPX, DBS.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC, Nasfroth.
• Dosis de espumante: No entregado.
• Otros reactivos: α-Fe_2O.
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Columna.
• Tamaño de partícula: No entregado.
• Velocidad gas: 1,5 cm/s.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 3 min.

Test 72:
• Recuperación: 66,54%.
• pH: No entregado.
• Tipo de colector: SBX.
• Dosis de colector: 240 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: T-92.
• Dosis de espumante: 120 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita, Otros.
• Tipo de celda: No entregado.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 73:
• Recuperación: 94,50%.
• pH: No entregado.
• Tipo de colector: SIBX, ADD.
• Dosis de colector: 120, 60 g/t.
• Tipo de depresante: Na_2S.
• Dosis de depresante: 100 g/t.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: 15 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 5 min.

Test 74:
• Recuperación: No entregado.
• pH: 10.
• Tipo de colector: No entregado.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: No entregado.
• Otros reactivos: NaCl, KCl, $MgSO_4$.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua de mar, Agua salina.
• Tipo de mineral: Calcopirita, Otros.
• Tipo de celda: No entregado.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 75:
• Recuperación: 82,22%.
• pH: No entregado.
• Tipo de colector: SBX, TC 1000, Aerofloat.
• Dosis de colector: No entregado.
• Tipo de depresante: Vidrio líquido.
• Dosis de depresante: No entregado.
• Tipo de espumante: T-92.
• Dosis de espumante: No entregado.
• Otros reactivos: Na_2S.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Relaves.
• Tipo de celda: No entregado.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 26 min.
• Tiempo de acondicionamiento: 10 min.

Test 76:
• Recuperación: 86,00%.
• pH: 9.
• Tipo de colector: SEX.
• Dosis de colector: 50 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: 25 g/t.
• Otros reactivos: Na_2CO_3.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 10 min.
• Tiempo de acondicionamiento: 3 min.

Test 77:
• Recuperación: 90,00%.
• pH: 9,5.
• Tipo de colector: BX, Aerofloat de butilo.
• Dosis de colector: 130 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: Agentes tecflote.

Test 78:
• Recuperación: No entregado.
• pH: 8,5.
• Tipo de colector: MX7017.
• Dosis de colector: 30 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.

• Dosis de espumante: No entregado.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita.
• Tipo de celda: No entregado.
• Tamaño de partícula: No entregado.
• Velocidad gas: 5,3 cm^3/min.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 3 min.

• Dosis de espumante: 15 g/t.
• Otros reactivos: Gasóleo.
• Dosis de otros reactivos: 15 g/t.
• Tipo de agua: Agua de mar.
• Tipo de mineral: Calcopirita, Bornita, Calcosina, Covelina, Cobre nativo.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 170 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 30 min.
• Tiempo de acondicionamiento: 1 min.

Test 79:
• Recuperación: 67,50%.
• pH: No entregado.
• Tipo de colector: PBX.
• Dosis de colector: 50 g/t.
• Tipo de depresante: Na_2S.
• Dosis de depresante: 50 g/t.
• Tipo de espumante: T-92.
• Dosis de espumante: 50 g/t.
• Otros reactivos: R-66.
• Dosis de otros reactivos: 100 g/t.
• Tipo de agua: Agua.
• Tipo de mineral: No entregado.
• Tipo de celda: No entregado.
• Tamaño de partícula: 0,071 - 2,0 mm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 5 min.

Test 80:
• Recuperación: 60,21%.
• pH: No entregado.
• Tipo de colector: PBX.
• Dosis de colector: 50 g/t.
• Tipo de depresante: Na_2S, Cal.
• Dosis de depresante: 50 g/t.
• Tipo de espumante: T-92.
• Dosis de espumante: 50 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: No entregado.
• Tipo de celda: No entregado.
• Tamaño de partícula: 0,071 - 2,0 mm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 5 min.

Test 81:
• Recuperación: 88,76%.
• pH: 9,3.
• Tipo de colector: PX.
• Dosis de colector: 1000 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: No entregado.
• Otros reactivos: HCl, NaOH.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua ultrapura.
• Tipo de mineral: Malaquita.
• Tipo de celda: No entregado.
• Tamaño de partícula: Menor a 2 µm.
• Velocidad gas: No entregado.

Test 82:
• Recuperación: 82,79%.
• pH: No entregado.
• Tipo de colector: BX.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: No entregado.
• Otros reactivos: Na_2CO_3.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 74 µm.
• Velocidad gas: No entregado.

• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: No entregado.

Test 83:
• Recuperación: 89,00%.
• pH: 9,5.
• Tipo de colector: Aerophine 3418A, Aero 3894.
• Dosis de colector: 60 g/t.
• Tipo de depresante: Cal, NaCN, Na_2SO_3, $Na_2S_2O_5$, Na_2S, $ZnSO_4$.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: 50 g/t.
• Otros reactivos: H_2SO_4.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua.
• Tipo de mineral: Calcopirita, Calcosina, Bornita, Covelina.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 40 µm.
• Velocidad gas: 2,5 dm^3/min.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 6 min.

Test 85:
• Recuperación: 94,00%.
• pH: 9,3.
• Tipo de colector: KAX, Aerophine 3418A.
• Dosis de colector: 120, 30 g/t.
• Tipo de depresante: Na_2SiO_3, $ZnSO_4$.
• Dosis de depresante: 7000, 1000 g/t.
• Tipo de espumante: MIBC, Dowfroth D250.
• Dosis de espumante: 90 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: Neumática.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 60 min.
• Tiempo de acondicionamiento: 20 min.

• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 84:
• Recuperación: 86,13%.
• pH: No entregado.
• Tipo de colector: NaIX.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: No entregado.
• Otros reactivos: $Na_2S{\cdot}9H_2O$, NH_4Cl, $CuSO_4{\cdot}5H_2O$.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua ultrapura.
• Tipo de mineral: Malaquita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: 38 - 74 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 20 min.

Test 86:
• Recuperación: 74,00%.
• pH: 9,3.
• Tipo de colector: KAX, Aerophine 3418A.
• Dosis de colector: 120, 30 g/t.
• Tipo de depresante: Na_2SiO_3, $ZnSO_4$.
• Dosis de depresante: 7000, 1000 g/t.
• Tipo de espumante: MIBC, Dowfroth D250.
• Dosis de espumante: 90 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado
• Tiempo de flotación: 60 min.
• Tiempo de acondicionamiento: 20 min.

Test 87:
• Recuperación: 92,18%.
• pH: 11.
• Tipo de colector: PAX.
• Dosis de colector: No entregado.
• Tipo de depresante: Cal.
• Dosis de depresante: No entregado.
• Tipo de espumante: Senfroth S522.
• Dosis de espumante: No entregado.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: No entregado.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 88:
• Recuperación: 91,00%.
• pH: 10.
• Tipo de colector: TC 1000.
• Dosis de colector: 200 g/t.
• Tipo de depresante: Cal.
• Dosis de depresante: No entregado.
• Tipo de espumante: Dowfroth D250.
• Dosis de espumante: 8 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 0.074 mm.
• Velocidad gas: 360 L/min.
• Tiempo de flotación: 40 min.
• Tiempo de acondicionamiento: 5 min.

Test 89:
• Recuperación: 84,80%.
• pH: No entregado.
• Tipo de colector: BX.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: No entregado.
• Otros reactivos: NH_4Cl, $CuSO_4 \cdot 5H_2O$, $Na_2S \cdot 9H_2O$.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua desionizada pura.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 5 min.
• Tiempo de acondicionamiento: 7 min.

Test 90:
• Recuperación: 75,00%.
• pH: 12.
• Tipo de colector: MIBC, Z11, C7240.
• Dosis de colector: No entregado.
• Tipo de depresante: A65.
• Dosis de depresante: No entregado.
• Tipo de espumante: No entregado.
• Dosis de espumante: No entregado.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua dulce, Agua reciclada.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 80 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 12 min.
• Tiempo de acondicionamiento: 3 min.

Test 91:
• Recuperación: 55,00%.
• pH: 9.
• Tipo de colector: KAX.
• Dosis de colector: 60 g/t.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.

Test 92:
• Recuperación: 72,79%.
• pH: 7.
• Tipo de colector: BX, Aerofloat de butilo.
• Dosis de colector: 50 g/t.
• Tipo de depresante: Na_2S.
• Dosis de depresante: No entregado.

• Tipo de espumante: Aceite de pino.
• Dosis de espumante: 50 g/t.
• Otros reactivos: H_2O_2.
• Dosis de otros reactivos: 300 g/t.
• Tipo de agua: Agua ultrapura.
• Tipo de mineral: Enargita, Calcopirita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 30 min.
• Tiempo de acondicionamiento: 60 min.

• Tipo de espumante: No entregado.
• Dosis de espumante: No entregado.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: Calcopirita, Cobre mate, Bornita.
• Tipo de celda: No entregado.
• Tamaño de partícula: Menor a 48 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 93:
• Recuperación: 83,58%.
• pH: No entregado.
• Tipo de colector: SBX, TC, Reaflot.
• Dosis de colector: No entregado.
• Tipo de depresante: Cal.
• Dosis de depresante: 0.75 kg/t.
• Tipo de espumante: T-92.
• Dosis de espumante: 85 g/t.
• Otros reactivos: Na_2S.
• Dosis de otros reactivos: 25 g/t.
• Tipo de agua: No entregado.
• Tipo de mineral: Calcopirita, Calcosina.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 74 µm.
• Velocidad gas: 3,1 L/min.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Test 94:
• Recuperación: 62,00%.
• pH: 8.
• Tipo de colector: PAX.
• Dosis de colector: 800 g/t.
• Tipo de depresante: Na_2S.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: 100 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 75 µm.
• Velocidad gas: 2,5 L/min.
• Tiempo de flotación: 8 min.
• Tiempo de acondicionamiento: 6 min.

Test 95:
• Recuperación: 66,95%.
• pH: No entregado.
• Tipo de colector: SEX, ADD, B201.
• Dosis de colector: No entregado.
• Tipo de depresante: Na_2SiO_3.
• Dosis de depresante: No entregado
• Tipo de espumante: No entregado.
• Dosis de espumante: No entregado.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: Calcopirita.

Test 96:
• Recuperación: 86.65%.
• pH: 10,5.
• Tipo de colector: EX.
• Dosis de colector: 100 g/t.
• Tipo de depresante: S-7261A.
• Dosis de depresante: 100 g/t.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: 60 g/t.
• Otros reactivos: Na_2SiO_3, Cal.
• Dosis de otros reactivos: 1200, 6000 g/t.
• Tipo de agua: Agua, Agua estancada de relaves.

• Tipo de celda: No entregado.
• Tamaño de partícula: Menor a 0,074 mm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 4 min.
• Tiempo de acondicionamiento: No entregado.

Test 97:
• Recuperación: 92,47%.
• pH: 9.
• Tipo de colector: SBX.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: MIBC.
• Dosis de espumante: No entregado.
• Otros reactivos: ST.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua desoinizada.
• Tipo de mineral: Malaquita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 74 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 2 min.
• Tiempo de acondicionamiento: No entregado.

Test 99:
• Recuperación: 95,41%.
• pH: No entregado.
• Tipo de colector: No entregado.
• Dosis de colector: No entregado.
• Tipo de depresante: No entregado.
• Dosis de depresante: No entregado.
• Tipo de espumante: Aceite de terpineol.
• Dosis de espumante: 80 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: Calcopirita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: No entregado.
• Tiempo de flotación: 3 min.
• Tiempo de acondicionamiento: 3 min.

• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: No entregado.
• Velocidad gas: 160 L/h.
• Tiempo de flotación: 2,5 min.
• Tiempo de acondicionamiento: 3 min.

Test 98:
• Recuperación: 92,89%.
• pH: 10.
• Tipo de colector: PAX.
• Dosis de colector: 300 g/t.
• Tipo de depresante: Na_2SiO_3.
• Dosis de depresante: 250 g/t.
• Tipo de espumante: Aceite de pino.
• Dosis de espumante: 250 g/t.
• Otros reactivos: No entregado.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: Agua de grifo.
• Tipo de mineral: No entregado.
• Tipo de celda: No entregado.
• Tamaño de partícula: 149 µm.
• Velocidad gas: No entregado.
• Tiempo de flotación: 10 min.
• Tiempo de acondicionamiento: 10 min.

Test 100:
• Recuperación: 62,36%.
• pH: No entregado.
• Tipo de colector: No entregado.
• Dosis de colector: No entregado.
• Tipo de depresante: Na_2S, MBS.
• Dosis de depresante: 160, 1 g/t.
• Tipo de espumante: No entregado.
• Dosis de espumante: No entregado.
• Otros reactivos: $ZnSO_4$.
• Dosis de otros reactivos: No entregado.
• Tipo de agua: No entregado.
• Tipo de mineral: No entregado.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 2 mm.
• Velocidad gas: No entregado.
• Tiempo de flotación: No entregado.
• Tiempo de acondicionamiento: No entregado.

Anexo C: Combinaciones aleatorias de parámetros.

Test 1:
- pH: 7.
- Tipo de colector: SIBX.
- Dosis de colector: 10 g/t.
- Tipo de depresante: Na_2SO_3.
- Dosis de depresante: 50 g/t.
- Tipo de espumante: Hydrofroth.
- Dosis de espumante: 10 g/t.
- Otros reactivos: SiO_2.
- Dosis de otros reactivos: 6 kg/t.
- Tipo de agua: Agua.
- Tipo de mineral: Relaves.
- Tipo de celda: Mecánica.
- Tamaño de partícula: Menor a 75 µm.
- Velocidad gas: 7 L/min.
- Tiempo de flotación: 2 min.
- Tiempo de acondicionamiento: 7 min.

Test 2:
- pH: 11.
- Tipo de colector: SIBX.
- Dosis de colector: 100 g/t.
- Tipo de depresante: Na_2SiO_3.
- Dosis de depresante: 150 g/t.
- Tipo de espumante: Hydrofroth.
- Dosis de espumante: 30 g/t.
- Otros reactivos: SiO_2.
- Dosis de otros reactivos: 4 kg/t.
- Tipo de agua: Agua de mar, Agua salina.
- Tipo de mineral: Calcopirita.
- Tipo de celda: Neumática.
- Tamaño de partícula: Menor a 170 µm
- Velocidad gas: 1,5 cm/s.
- Tiempo de flotación: 8 min.
- Tiempo de acondicionamiento: 3 min.

Test 3:
- pH: 12.
- Tipo de colector: SIBX.
- Dosis de colector: 80 g/t.
- Tipo de depresante: Na_2S.
- Dosis de depresante: 100 g/t.
- Tipo de espumante: MIBC, Nasfroth.
- Dosis de espumante: No añadido.
- Otros reactivos: SiO_2.
- Dosis de otros reactivos: 5 kg/t.
- Tipo de agua: Agua de mar.
- Tipo de mineral: Calcopirita, Otros.
- Tipo de celda: Columna.
- Tamaño de partícula: 0,071 - 2,0 mm.
- Velocidad gas: 5,3 cm^3/min.
- Tiempo de flotación: 3 min.
- Tiempo de acondicionamiento: 5 min.

Test 4:
- pH: 10.
- Tipo de colector: SIPX, DBS.
- Dosis de colector: No añadido.
- Tipo de depresante: Vidrio líquido.
- Dosis de depresante: No añadido.
- Tipo de espumante: T-92.
- Dosis de espumante: 120 g/t.
- Otros reactivos: TiO_2
- Dosis de otros reactivos: 6 kg/t.
- Tipo de agua: Agua ultrapura.
- Tipo de mineral: Calcopirita, Bornita, Calcocina, Covelina, Cobre nativo.
- Tipo de celda: Mecánica.
- Tamaño de partícula: 0,071 - 2,0 mm.
- Velocidad gas: 2.5 dm^3/min.
- Tiempo de flotación: 26 min.
- Tiempo de acondicionamiento: 10 min.

Test 5:
- pH: 9.
- Tipo de colector: SIPX, DBS.
- Dosis de colector: No añadido.
- Tipo de depresante: Na_2S.
- Dosis de depresante: 50 g/t.
- Tipo de espumante: Aceite de terpineol.
- Dosis de espumante: 15 g/t.

Test 6:
- pH: 9,5.
- Tipo de colector: SIPX, DBS.
- Dosis de colector: No añadido.
- Tipo de depresante: Na_2S, Cal.
- Dosis de depresante: 50 g/t.
- Tipo de espumante: MIBC.
- Dosis de espumante: 25 g/t.

• Otros reactivos: TiO_2.
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua desionizada pura.
• Tipo de mineral: Malaquita.
• Tipo de celda: Neumática.
• Tamaño de partícula: Menor a 2 µm.
• Velocidad gas: 360 L/min.
• Tiempo de flotación: 10 min.
• Tiempo de acondicionamiento: 1 min.

• Otros reactivos: TiO_2
• Dosis de otros reactivos: 5 kg/t.
• Tipo de agua: Agua dulce, Agua reciclada.
• Tipo de mineral: Calcopirita, Calcosina, Bornita, Covelina.
• Tipo de celda: Columna.
• Tamaño de partícula: Menor a 74 µm.
• Velocidad gas: 3,1 L/min.
• Tiempo de flotación: 30 min.
• Tiempo de acondicionamiento: 6 min.

Test 7:
• pH: 8,5.
• Tipo de colector: SBX.
• Dosis de colector: 240 g/t.
• Tipo de depresante: Cal, NaCN, Na_2SO_3, $Na_2S_2O_5$, Na_2S, $ZnSO_4$.
• Dosis de depresante: No añadido.
• Tipo de espumante: Agentes tecflote.
• Dosis de espumante: No añadido.
• Otros reactivos: γ-Al_2O_3.
• Dosis de otros reactivos: 6 kg/t.
• Tipo de agua: Agua, Agua estancada de relaves.
• Tipo de mineral: Enargita, Calcopirita.
• Tipo de celda: Mecánica.
• Tamaño de partícula: Menor a 40 µm.
• Velocidad gas: 2,5 L/min.
• Tiempo de flotación: 60 min.
• Tiempo de acondicionamiento: 20 min.

Test 8:
• pH: 9,3.
• Tipo de colector: SBX, TC 1000, Aerofloat.
• Dosis de colector: 120 g/t.
• Tipo de depresante: Na_2SiO_3, $ZnSO_4$.
• Dosis de depresante: 7000, 1000 g/t.
• Tipo de espumante: MIBC.
• Dosis de espumante: 15 g/t.
• Otros reactivos: γ-Al_2O_3
• Dosis de otros reactivos: 4 kg/t.
• Tipo de agua: Agua desionizada.
• Tipo de mineral: Calcopirita, Cobre mate, Bornita.
• Tipo de celda: Neumática.
• Tamaño de partícula: 38 - 74 µm.
• Velocidad gas: 160 L/h.
• Tiempo de flotación: 40 min.
• Tiempo de acondicionamiento: 60 min.

Anexo D: Datos Figura 10.

Recuperación Cu Real	Recuperación Cu Predicción	pH
54,00%	48,19%	10
51,00%	52,28%	10
55,80%	53,96%	10
54,20%	57,17%	10
60,00%	55,07%	10
54,50%	56,27%	10
57,00%	57,00%	10
50,00%	54,08%	10
55,00%	53,94%	10
51,20%	52,16%	10

54,00%	54,31%	10
52,90%	53,41%	10
53,70%	53,07%	10
52,00%	51,34%	10
54,20%	53,85%	10
58,00%	57,53%	10
53,00%	53,30%	10
55,60%	55,63%	10
54,00%	55,58%	10
51,50%	53,48%	10
62,00%	85,73%	11
60,00%	25,62%	11
65,00%	52,73%	11
68,20%	62,26%	11
60,00%	58,46%	11
62,00%	58,92%	11
61,00%	59,48%	11
64,50%	63,00%	11
65,20%	63,20%	11
62,50%	60,38%	11
60,00%	60,28%	11
58,10%	60,03%	11
64,00%	62,12%	11
64,90%	63,52%	11
62,60%	62,03%	11
64,00%	62,06%	11
63,00%	62,39%	11
64,00%	62,56%	11
62,00%	63,83%	11
59,00%	60,43%	11
62,60%	63,08%	11
62,00%	63,37%	11
63,00%	62,93%	11
60,30%	59,97%	11
64,80%	64,82%	11
62,70%	62,95%	11
62,00%	65,38%	11
70,00%	49,19%	12
52,80%	60,34%	12
65,00%	62,44%	12
64,00%	60,17%	12
55,10%	62,33%	12
67,00%	64,13%	12
64,50%	62,16%	12
69,20%	67,83%	12

54,10%	64,49%	12
72,00%	66,31%	12
63,00%	63,85%	12
53,00%	55,72%	12
65,00%	65,79%	12
70,00%	68,92%	12
67,00%	63,99%	12
63,20%	63,15%	12
63,00%	63,14%	12
63,20%	63,28%	12
55,00%	59,41%	12
64,00%	61,55%	12