



# **Práctica 2:** **Clasificación de minerales**

CE Inteligencia Artificial y Big Data  
Modelos de Inteligencia Artificial  
2024/2025

Daniel Marín López  
Guadalupe Luna Velázquez

## Índice

1. Problema.....	3
2. Características a tener en cuenta .....	4
3. Definición del problema en PROLOG.....	4
4. Reglas de nuestro sistemas (Problema principal).....	5
5. Conclusión .....	10
6. Webgrafía.....	10

## 1. Problema

Se nos plantea el siguiente problema para un sistema experto:

*“Dadas las características de un mineral (por ejemplo, forma, color, dureza,...), el sistema debe indicar el nombre de dicho mineral. El número mínimo de características a considerar debe ser 5, y el número de minerales diferentes que tenga predefinido el sistema debe ser de 25.”*

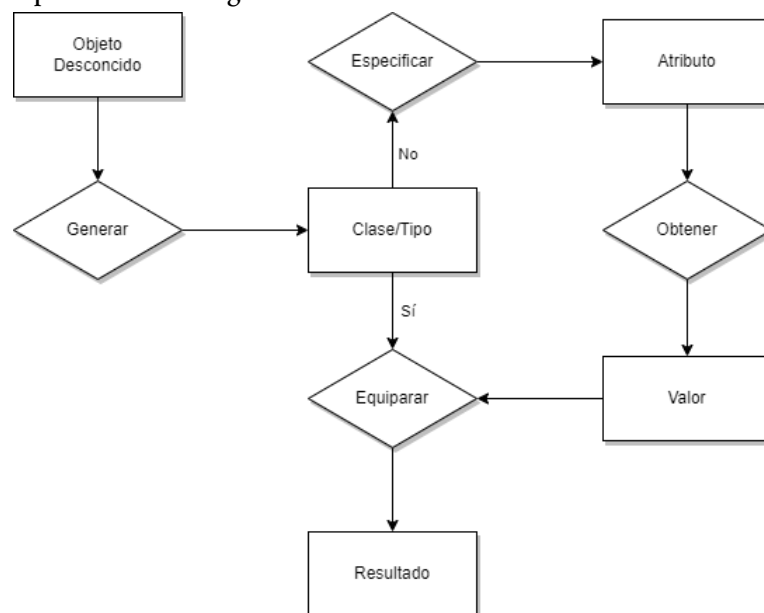
Este problema es un típico problema de **clasificación**, en donde tenemos una serie de características y un objetivo o etiqueta que corresponde a esas características. En el caso de un sistema experto debe reconocer objetos desconocidos para él e identificarlo como miembro de una clase conocida, para ello se deben tener en cuenta dos cosas:

- **Dominio de la clase conocida**
- **Características que tiene ese objeto desconocido**

Este problema no es el único, este problema también se presenta a la hora de descubrir nuevas especies de animales o plantas. La terminología que hemos usado es la siguiente:

- **Objeto:** Aquello desconocido para el sistema que queremos clasificar.
- **Clase:** Categoría que comparten varios objetos con varias similitudes.
- **Atributo:** Propiedades usadas para identificar las clases.
- **Característica:** Propiedades verificadas por un valor de un objeto.

Podríamos representar este problema de la siguiente manera:



Si pensamos en nuestro problema, lo que haremos será indicar las características de un mineral X que no conocemos y con eso tendrá que comprobar con los datos introducidos anteriormente. Al final, se imprimirá el nombre del mineral en pantalla. Para este sistema experto usaremos el lenguaje PROLOG en la página SWISH que nos ejecuta automáticamente el código de manera fácil e intuitiva.

## 2. Características a tener en cuenta

El sistema considerará un mineral aquel que tenga las siguientes características:

- **Nombre:** Nombre del mineral.
- **Color:** Color del mineral.
- **Dureza:** Rango que puede tener la dureza del mineral.
- **Fractura:** Como está fracturado el mineral. Tenemos los siguientes valores:
  - **Irregular:** Cuando un mineral se rompe según superficies bastas e irregulares.
  - **Astilosa:** Cuando un mineral se rompe se vuelve fibras o astillas.
  - **Concoidea:** Cuando la fractura tiene superficies suaves, lisas, como la cara interior de una concha. El término **subconcoidea** también está incluido aquí.
  - **Escamosa:** Aquella cuya superficie es tosca e irregular, y muestra bordes agudos y dentados.
  - **Cuadrangular:** Por la cual se pueden separar presentando superficies planas y paralelas a las caras reales.
- **Transparencia:** La transparencia que tiene el mineral.
- **Sistema Cristalino:** Cómo se forma el mineral. Tenemos los siguientes valores:
  - **Cúbico/Isométrico:** Se caracteriza porque la celda unidad de la red cristalina tiene la forma geométrica de cubo, ya que tiene los tres ángulos rectos y las tres aristas de la celda iguales.
  - **Ortorrómbico:** Se caracteriza porque la celda unidad de la red cristalina tiene la forma geométrica con los tres ángulos rectos, mientras que las tres aristas de dicha celda unidad tienen todas longitudes diferentes.
  - **Hexagonal:** Tiene la misma simetría que un prisma regular con una base hexagonal.
  - **Monoclínico:** Consta de un eje binario, un plano perpendicular a éste y un centro de inversión.
  - **Trigonal/Romboedrica:** Para algunos autores no es considerado un sistema cristalino, sino una variante dentro del sistema cristalino hexagonal
  - **Poroso:** No aparece como sistema, pero si es una variable para uno de los almacenados.

## 3. Definición del problema en PROLOG

Para crear un sistema en PROLOG para clasificar minerales debemos definir una **base de conocimiento** que usará nuestro sistema para clasificar en base a unas **reglas** establecidas para la correcta identificación de un mineral. Un mineral lo definiremos de la siguiente manera:



```
mineral(nombre, color, dureza, fractura, transparencia, sistema_cristalino)
```

Con esto, definimos la base de conocimiento de nuestro sistema que tendrá un total de 25 minerales:

```
mineral("Cobre nativo", "Rosa", "2.5-3", "Irregular", "Opaca", "Isometrico").
mineral("Plata", "Blanco grisaceo", "2.5-3", "Astillosa", "Opaca", "Cubico").
mineral("Aragonita", "Marron", "3.5-4", "Concoidea", "Translucido", "Ortorrombico").
mineral("Calcita Roja", "Rojo", "3", "Concoidea", "Translucido", "Hexagonal").
mineral("Malaquita", "Verde", "3.5-4", "Concoidea", "Opaca", "Monoclinico").
mineral("Smithsonita", "Variado", "4-4.5", "Irregular", "Translucido", "Trigonal").
mineral("Fluorapatito", "Incoloro", "5", "Concoidea", "Transparente", "Hexagonal").
mineral("Mimetita", "Variado", "3.5-4", "Subcocoidea", "Translucido", "Hexagonal").
mineral("Piromorfita", "Variado", "3.5-4", "Subcocoidea", "Transparente", "Hexagonal").
mineral("Fluorita", "Variado", "4", "Cuadrangular", "Translucido", "Cubico").
mineral("Cuarzo", "Blanco", "7", "Concoidea", "Transparente", "Trigonal").
mineral("Barita", "Variado", "3-3.5", "Irregular", "Transparente", "Ortorrombico").
mineral("Selenita", "Incoloro", "1.5-2", "Concoidea", "Transparente", "Monoclinico").
mineral("Bornita", "Rojo", "3-3.5", "Concoidea", "Opaca", "Tetragonal").
mineral("Esfalerita", "Amarillo", "3.5-4", "Concoidea", "Opaca", "Cubico").
mineral("Arsenico", "Gris", "3.5", "Irregular", "Opaca", "Romboedrica").
mineral("Azufre", "Amarillo", "2", "Concoidea", "Opaca", "Ortorrombico").
mineral("Bauxita", "Rojo", "1-3", "Concoidea", "Opaca", "Poroso").
mineral("Bismuto", "Rojo", "2.25", "Irregular", "Opaca", "Romboedrica").
mineral("Calcopirita", "Amarillo", "3.5-4", "Concoidea", "Opaca", "Tetragonal").
mineral("Cromatita", "Amarillo", "2", "Concoidea", "Translucido", "Tetragonal").
mineral("Diamante", "Variado", "10", "Concoidea", "Translucido", "Cubico").
mineral("Esmeralda", "Verde", "7.5-8", "Concoidea", "Translucido", "Hexagonal").
mineral("Grafito", "Gris", "1-2", "Escamosa", "Opaca", "Hexagonal").
mineral("Magnesita", "Blanco", "4-4.5", "Concoidea", "Transparente", "Trigonal").
mineral("Rubi", "Rojo", "9", "Concoidea", "Translucido", "Trigonal").
```

#### 4. Reglas de nuestro sistemas (Problema principal)

Cómo hemos indicado anteriormente, nuestro sistema se encargará de clasificar/identificar un mineral en base a las características que le pase el usuario por pantalla. Para ello podemos hacer dos funciones:

- La primera función pedirá al usuario los valores necesarios para identificar un mineral, si se detecta que el usuario no escribió nada entonces saltará un error diciendo que el campo no puede estar vacío y empezará de nuevo desde el inicio.
- La segunda función se encargará de mirar en la base de conocimiento si hay un mineral que cumpla con todas las características indicadas por el usuario anteriormente. Si lo encuentra devolverá el nombre del mineral y si no devolverá un mensaje indicando que no hay un mineral con esas propiedades.

```
identificar_mineral(Color, Dureza, Fractura, Transparencia, Sistema):-
    ( mineral(Nombre, Color, Dureza, Fractura, Transparencia, Sistema) ->
        format('El mineral es: ~w~n', [Nombre])
    ;
        writeln('No se pudo identificar el mineral')
    ).
```

Velázquez

```

clasificar_mineral :-
    repeat,
    write('Introduce el color del mineral: '),
    read_line_to_string(user_input, Color),
    (Color == "" -> write('¡El color no puede estar vacío! Intenta de nuevo. '), nl, fail ;
true),

    write('Introduce la dureza del mineral: '),
    read_line_to_string(user_input, Dureza),
    (Dureza == "" -> write('¡La dureza no puede estar vacía! Intenta de nuevo. '), nl, fail ;
true),

    write('Introduce la fractura del mineral: '),
    read_line_to_string(user_input, Fractura),
    (Fractura == "" -> write('¡La fractura no puede estar vacía! Intenta de nuevo. '), nl,
fail ; true),

    write('Introduce la transparencia del mineral: '),
    read_line_to_string(user_input, Transparencia),
    (Transparencia == "" -> write('¡La transparencia no puede estar vacía! Intenta de
nuevo. '), nl, fail ; true),

    write('Introduce el sistema cristalino del mineral: '),
    read_line_to_string(user_input, Sistema),
    (Sistema == "" -> write('¡El sistema no puede estar vacío! Intenta de nuevo. '), nl, fail
; true),

    % Mostrar los valores ingresados
    format('~nResumen de entrada:~nColor: ~w~nDureza: ~w~nFractura: ~w~nTransparencia:
~w~nSistema: ~w~n~n',
        [Color, Dureza, Fractura, Transparencia, Sistema]),

    identificar_mineral(Color, Dureza, Fractura, Transparencia, Sistema),
    !.

```

Introduce la dureza del mineral:	Introduce la dureza del mineral:
9	8
Introduce la fractura del mineral:	Introduce la fractura del mineral:
Concoidea	Concoidea
Introduce la transparencia del mineral:	Introduce la transparencia del mineral:
Translucido	Translucido
Introduce el sistema cristalino del mineral:	Introduce el sistema cristalino del mineral:
Trigonal	Trigonal
Resumen de entrada:	Resumen de entrada:
Color: Rojo	Color: Rojo
Dureza: 9	Dureza: 8
Fractura: Concoidea	Fractura: Concoidea
Transparencia: Translucido	Transparencia: Translucido
Sistema: Trigonal	Sistema: Trigonal
El mineral es: Rubi	No se pudo identificar el mineral
true	true

Velázquez

Con esto, el problema principal de la clasificación estaría resuelto. Y partiendo de esta base, se puede mejorar el sistema para que hiciera otras tareas que lo enriquezcan como listar los minerales en base a una característica o varias. También se podría crear un menú para que el usuario pueda realizar las tareas de manera más fácil y teniendo todo unificado. Algo que hemos implementado a continuación.

Además se ha creado la función para que liste las características de un mineral a partir del nombre, también nos avisará en este caso si el nombre está vacío dándonos un error (teniendo que volver a introducirlo) o si no encuentra el mineral, lo avisará por pantalla sin ningún resultado.

```
caracteristicas_mineral :-
    repeat,
    write('Introduce el nombre del mineral: '),
    read_line_to_string(user_input, Nombre_buscado),
    (Nombre_buscado == "" ->
        write(';El nombre no puede estar vacío! Intenta de nuevo. '), nl, fail
    );
    ( mineral(Nombre_buscado, Color, Dureza, Fractura, Transparencia, Sistema) ->
        format('~nCaracterísticas de ~w~n', [Nombre_buscado]),
        format('Color: ~w~n', [Color]),
        format('Dureza: ~w~n', [Dureza]),
        format('Fractura: ~w~n', [Fractura]),
        format('Transparencia: ~w~n', [Transparencia]),
        format('Sistema cristalino: ~w~n', [Sistema]),
        !
    );
    writeln('Mineral no encontrado. '), !
).

```

 **caracteristicas\_mineral.**

Introduce el nombre del mineral:

*Malaquita*

Características de Malaquita:

Color: Verde

Dureza: 3.5-4

Fractura: Concoidea

Transparencia: Opaca

Sistema cristalino: Monoclinico

**true**

?-

*caracteristicas\_mineral.*

Velázquez

Por último, se ha creado la función para listar los minerales que coincidan con las características que el usuario diga, como la primera función pero sin ser necesario escribir todas las características y dando como resultado una lista de los nombres de los minerales que cumplan esas características dichas.

```

buscar_caracteristicas:-
    repeat,
    write('Introduce el color del mineral (o deja en blanco): '),
    read_line_to_string(user_input, Color),

    write('Introduce la dureza del mineral (o deja en blanco): '),
    read_line_to_string(user_input, Dureza),

    write('Introduce la fractura del mineral (o deja en blanco): '),
    read_line_to_string(user_input, Fractura),

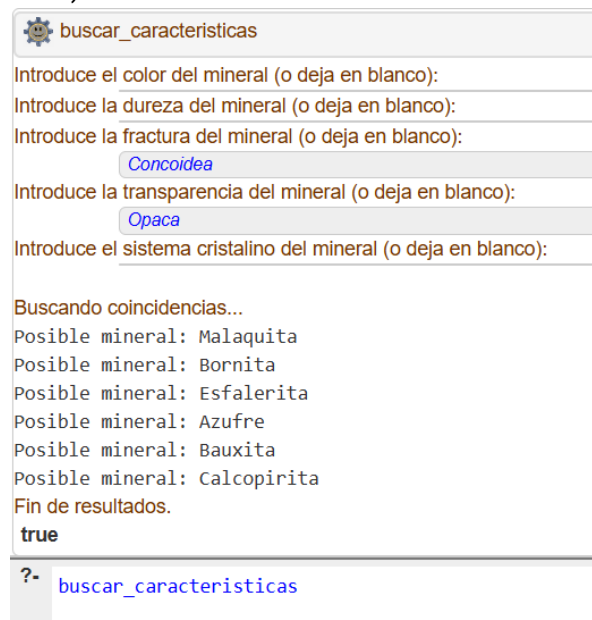
    write('Introduce la transparencia del mineral (o deja en blanco): '),
    read_line_to_string(user_input, Transparencia),

    write('Introduce el sistema cristalino del mineral (o deja en blanco): '),
    read_line_to_string(user_input, Sistema),

    nl, writeln('Buscando coincidencias...'),

    ( mineral(Nombre, ColorM, DurezaM, FracturaM, TransparenciaM, SistemaM),
      (Color == "" ; Color == ColorM),
      (Dureza == "" ; Dureza == DurezaM),
      (Fractura == "" ; Fractura == FracturaM),
      (Transparencia == "" ; Transparencia == TransparenciaM),
      (Sistema == "" ; Sistema == SistemaM),
      format('Posible mineral: ~w~n', [Nombre]),
      fail
    );
    writeln('Fin de resultados.'), !
).

```



```

?- buscar_caracteristicas

```

Introduce el color del mineral (o deja en blanco):

Introduce la dureza del mineral (o deja en blanco):

Introduce la fractura del mineral (o deja en blanco):

Concoidea

Introduce la transparencia del mineral (o deja en blanco):

Opaca

Introduce el sistema cristalino del mineral (o deja en blanco):

Buscando coincidencias...

Posible mineral: Malaquita

Posible mineral: Bornita

Posible mineral: Esfalerita

Posible mineral: Azufre

Posible mineral: Bauxita

Posible mineral: Calcopirita

Fin de resultados.

true

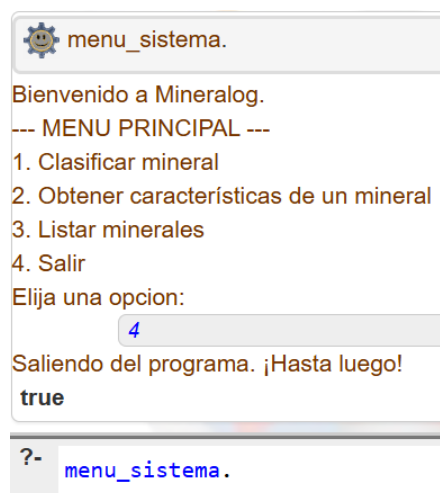


Para nuestro menú hemos implementado el siguiente código:

```
menu_sistema:-
    writeln('Bienvenido a Mineralog.'),
    writeln('--- MENU PRINCIPAL ---'),
    writeln('1. Clasificar mineral'),
    writeln('2. Obtener características de un mineral'),
    writeln('3. Listar minerales'),
    writeln('4. Salir'),
    write('Elija una opcion: '),
    read(Opcion),
    ejecutar_opcion(Opcion).

ejecutar_opcion(1) :-
    clasificar_mineral,
    menu_sistema.
ejecutar_opcion(2) :-
    características_mineral,
    menu_sistema.
ejecutar_opcion(3) :-
    buscar_características,
    menu_sistema.
ejecutar_opcion(4) :-
    writeln('Saliendo del programa. ¡Hasta luego!'),
    !.
ejecutar_opcion(_) :-
    writeln('Opción no válida. Intente de nuevo.'),
    menu_sistema.
```

De esta manera el usuario podrá acceder a las funciones sin necesidad de tener que escribir las funciones una a una, lo que hace que sea más accesible para los usuarios y que simplemente si se desea agregar otra función es tan fácil como editar el menú.



```
?- menu_sistema.
Bienvenido a Mineralog.
--- MENU PRINCIPAL ---
1. Clasificar mineral
2. Obtener características de un mineral
3. Listar minerales
4. Salir
Elija una opcion:
4
Saliendo del programa. ¡Hasta luego!
true
```

## 5. Conclusión

Tras partir de un simple problema como es la clasificación/identificación de minerales, hemos conseguido no solo el objetivo principal para realizar esta clasificación sino que también hemos implementado otras funciones que enriquecen nuestro sistema y un menú interactivo para los usuarios.

Si PROLOG diera cabida a la actualización de la base de conocimiento podríamos agregar más funciones como agregar un nuevo mineral, editarlo y eliminarlo de la base. Lo que haría que nuestro sistema tuviera la posibilidad de actualizar su propia base de datos sin tener que incluir nuevos minerales directamente en el código.

## 6. Webgrafía

1. [Fractura de un mineral: Wikipedia](#)
2. [Tipos de sistemas de los minerales: Wikipedia](#)