RESUMEN

Los análisis de procedencia, concretamente aquellos enfocados a la caracterización de fuentes y muestras, presentan un amplio abanico de aplicaciones entre las que destacan las histórico-humanistas, geológicas y químicas. Permiten ampliar el conocimiento sobre territorios antiguos, interacciones entre diferentes culturas y civilizaciones o sobre sus posibles rutas comerciales.

Este proyecto se propone como objetivo principal continuar el desarrollo de un programa informático que permite realizar análisis de procedencia basados en el estudio geoquímico de la obsidiana. Específicamente, se basa en el desarrollo de software estadístico en Python para el análisis de material arqueológico de Oriente Próximo y Medio.

El presente escrito se centra en el estudio y desarrollo de nuevas herramientas estadísticas implementadas en el software y en el posterior análisis de procedencia de piezas provenientes de un yacimiento. De este modo, se comprueba la validez y robustez de las diferentes técnicas aplicadas. Se cubre también la documentación detallada del programa, tanto a nivel de usuario mediante un tutorial de utilización, como a nivel de futuros programadores que decidan continuar con su desarrollo mediante documentación técnica.

SUMARIO

1. Glosario

Análisis discriminante: Técnica estadística multivariante que permite la clasificación de una muestra en diversos grupos conocidos de antemano en función de las propiedades que tiene.

Análisis componentes principales (ACP): Técnica estadística multivariante que permite reducir la dimensión de un conjunto de datos perdiendo el mínimo de información.

Anatolia: Península ocupada por la parte asiática de Turquía. Limita al norte con el mar Negro, al sur con el mar Mediterráneo, al oeste con el mar Egeo y al este con el resto del continente asiático.

Clase (programación): En programación orientada a objetos, tipo de datos definido por el usuario que especifica un conjunto de objetos que comparten las mismas características. Cada clase es un modelo que define un conjunto de variables o atributos y métodos apropiados para operar con dichos datos.

Coeficiente *PseudoF:* Parámetro o valor que pretende definir lo ajustado que están los grupos o conglomerados de un conjunto de datos. Esencialmente es la relación entre la media de la suma de cuadrados entre grupos (BGSS) y dicha media dentro de cada grupo (WGSS).

Datos heterogéneos: Conjuntos de datos de estructura diferente; no se han analizado las mismas propiedades y, por tanto, cada subconjunto presenta elementos diversos.

Datos homogéneos: Conjuntos de datos de estructura similar; se han analizado las mismas propiedades y, por tanto, cada subconjunto presenta los mismos elementos.

Diálogo (programación): Término utilizado a lo largo del trabajo para referirse a la ventana (*widget*) con la que interactúa el usuario final.

Estadística bivariante: Comprende herramientas estadísticas que estudian el comportamiento de dos variables al mismo tiempo.

Estadística multivariante: Comprende herramientas estadísticas que estudian el comportamiento de tres o más variables al mismo tiempo.

Heredar (programación): En programación orientada a objetos, mecanismo a través del cual es posible crear nuevas clases partiendo de una clase o de una jerarquía de

clases preexistente, extendiendo así su funcionalidad. La clase de la que se hereda suele denominarse clase padre.

Interfaz de usuario: Medio por el que el usuario es capaz de comunicarse con una máquina, equipo o computador que comprende todos los puntos de contacto entre el usuario y el equipo.

Interfaz gráfica de usuario (GUI, *Graphical User Interface*): Programa informático que actúa de interfaz de usuario utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Proporciona un entorno visual sencillo para permitir la comunicación con el sistema operativo o con el ordenador.

Biblioteca (programación): Conjunto de implementaciones funcionales que ofrece una interfaz bien definida para su uso. Habitualmente se emplea el término librería para referirse a una biblioteca.

Minería de datos: Campo de las ciencias de computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos. Utiliza métodos de inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos.

Objeto (programación): En programación orientada a objetos, es el resultado de instanciar una clase. Consta de un estado y de un comportamiento y adquiere la capacidad de interactuar con otros objetos.

Valor perdido: Valor que no fue recogido durante el proceso de medición.

Widget: En el contexto de la programación de aplicaciones visuales, los *widgets* tienen un significado amplio como componentes o control visual que implementa el programador. De él, emanan la mayoría de componentes gráficos.

Yacimiento (arqueológico): Concentración de restos arqueológicos constituido por la presencia de artefactos, elementos estructurales, suelos de ocupación y otra serie de anomalías.

2. Origen del proyecto y motivaciones

El planteamiento inicial del trabajo surgió de la propuesta de los profesores Francisco Javier Giménez y Lluís Solano, publicada junto al resto de temas en la bolsa de proyectos de la Escuela Técnica Superior de Ingeniería Industrial de Barcelona (ETSEIB). Este proyecto es la continuación del trabajo (1) realizado por Mauricio Andrés Alva Howes.

El proyecto nació de la necesidad de realizar un estudio de procedencia basado en la composición química de una serie de muestras de obsidiana. La obsidiana (o vidrio volcánico) es una roca ígnea volcánica que presenta una composición química diferente según su fuente de procedencia. Dadas la condiciones del estudio, el proyectista anterior concluyó que la solución ideal era la de implementar un programa informático que permitiese llevar a cabo, de la manera más simplificada posible, estos estudios.

El presente trabajo recoge el relevo de desarrollo de este programa, ampliando sus herramientas estadísticas para realizar estudios de procedencia con el mayor grado de exactitud posible. La principal motivación fue la de introducir técnicas que permitiesen cuantificar el grado de certeza de procedencia de una muestra determinada o el cálculo de probabilidades referentes al estudio. Para ello, el director del proyecto proporcionó información y material arqueológico proveniente de diferentes artículos (1-7) de Oriente Próximo y Medio. También se facilitaron propuestas y vías de continuación y mejora para el nuevo desarrollo.

A pesar del uso de un conjunto de datos determinado durante todo el proyecto, interesaba realizar una herramienta estadística extensible a otros juegos de datos.

Por último, este trabajo también tiene la intención de ser continuado por otros desarrolladores, ya sea aumentando las herramientas estadísticas implementadas o su ampliación para el estudio de otros materiales.

3. Introducción

La **obsidiana**, también conocida como vidrio volcánico, es una roca ígnea volcánica perteneciente al grupo de los silicatos que se forma en la última fase de la erupción volcánica. Su estructura atómica desordenada y químicamente inestable y poco definida hace que sea clasificada comúnmente como mineraloide.

Fue muy utilizada en gran parte del Mediterráneo y en el resto del mundo, desde Asia hasta América. Su alta dureza y facilidad para ser laminada favorece su uso en un amplio abanico de aplicaciones; desde la realización de herramientas u ornamentos en el pasado hasta su utilización en bisturís quirúrgicos en la actualidad. Fue utilizada también como piedra preciosa por su aspecto y por la variedad de colores que puede presentar.

El limitado número de fuentes de origen y composición química característica en cada una de ellas, hacen de la obsidiana un material idóneo para la aplicación de estudios de procedencia.

Los **estudios de procedencia**, junto a los estudios de datación, forman parte de los campos más importantes en los que se comprueba la utilidad de la química en ramas más humanistas, como la arqueología o la historia. Uno de los primeros estudios de procedencia basados en la composición química que se desarrollaron tuvo la obsidiana como objeto de estudio. A pesar de ello, es aún un terreno en el que queda bastante por indagar.

Estos estudios están basados en la existencia de algunas propiedades químicas, geológicas o geoquímicas, que difieren según la fuente de procedencia. La concentración de uno o varios elementos químicos que componen el material puede ser una de estas propiedades características.

Su funcionalidad se halla en el hecho de poder localizar el origen de una muestra o yacimiento del material desconocido a priori. Para ello, se realiza en primer lugar la caracterización de las fuentes a partir de la identificación de los elementos característicos o combinaciones de los mismos y, a posteriori, un contraste de estos con la muestra a estudiar. En caso de darse coincidencias inequívocas, se considera que dicha fuente es la de origen. Disponer de conocimientos en el campo de estudio puede resultar fundamental a la hora de cerciorar la fuente correctamente.

Los resultados obtenidos permiten ampliar el conocimiento sobre territorios antiguos, interacciones entre diferentes culturas y civilizaciones o sobre sus posibles rutas comerciales.

3.1. Objetivos del proyecto

El objetivo principal del proyecto radica en la continuación de desarrollo de una herramienta informática que permite la realización de estudios de análisis de procedencia. Estos estudios están basados en la composición química o mineralógica de los materiales.

Se puede desglosar el objetivo del proyecto en dos partes principales:

- Caracterización de subgrupos dentro de una fuente.
- Caracterización de yacimientos.

El primero de los objetivos supone una vía continuación del proyecto totalmente nueva. Esta vía debe permitir una caracterización de los subgrupos por los que puede estar formada una determinada fuente. Para ello, se analizarán diferentes algoritmos de agrupamiento.

La segunda parte consiste en seguir desarrollando técnicas para el análisis de procedencia de la obsidiana. En el proyecto anterior se desarrollaron técnicas que permitían caracterizar una fuente a partir de ciertos componentes químicos simples o fraccionales. En esta nueva etapa del proyecto se deberán añadir nuevas herramientas que permitan cuantificar el grado de certeza de la procedencia de una muestra determinada o el cálculo de probabilidades referentes al estudio. Por ello, el proyectista anterior propone diferentes algoritmos de clasificación.

3.2. Alcance del proyecto

Como ya se ha comentado, es conocida la intención de continuar con el desarrollo del proyecto a través de otras personas, aumentado así el contenido y la calidad del mismo.

El presente proyecto deberá abarcar los siguientes aspectos:

- Implementar nuevos algoritmos de clasificación.
- Implementar algoritmos de agrupamiento.

- Mejora en la representación gráfica de los datos.
- Ampliación de la interfaz gráfica.
- Redactado de documentación técnica que permita continuar su desarrollo.
- Redactado del manual de usuario.
- Descripción de las posibles vías de continuación.

4. Desarrollo del trabajo

4.1. Análisis previo

El primer punto de trabajo se centró en la lectura detallada de la memoria del proyectista anterior y en la exploración del estado de desarrollo del programa informático. También se dedicó un tiempo al análisis de las diferentes vías de continuación propuestas y al estudio del conjunto datos del que se disponía para realizar los estudios. Finalmente, se realizó una primera aproximación al código escrito para familiarizarse con la estructura que se había seguido hasta el momento.

4.1.1 Inmersión en el análisis de procedencia

El presente trabajo se centra en el análisis de material arqueológico de Oriente Próximo y Medio basándose en tres fuentes de procedencia: Göllü Dağ, Bingöl y Nemrut Dağ. Dada la imposibilidad de realizar el muestreo y posterior análisis químico, la fuente principal de datos se basa en una serie de artículos que fueron proporcionados por el director del trabajo. Aunque ya se realizó, por parte del proyectista anterior, un extenso estudio de estos artículos, se hace un pequeño resumen a modo de contextualización.

(2) Este artículo de origen francés supone un primer acercamiento a los métodos de análisis de procedencia. El estudio se realiza sobre la obsidiana iraquí de Tell Magzalia, Tell Sotto, Yarim Tepe y Kül Tepe y utiliza como métodos de caracterización de las fuentes el graficado bidimensional por grupos y el cálculo de funciones discriminantes.

(3) En este segundo artículo se realiza un estudio interno del volcán Göllü Dağ debido a su gran variabilidad. Se llevó a cabo mediante la recolección de muestras por ubicación (véase Figura 4.1). Las causas de tal variabilidad se sospechaban que venían dadas por distintos tipos de obsidiana a nivel interno, por la forma de su creación o el momento y factores involucrados durante el mismo. Los métodos empleados en este estudio fueron principalmente la representación bidimensional de elementos y el análisis de componentes principales.

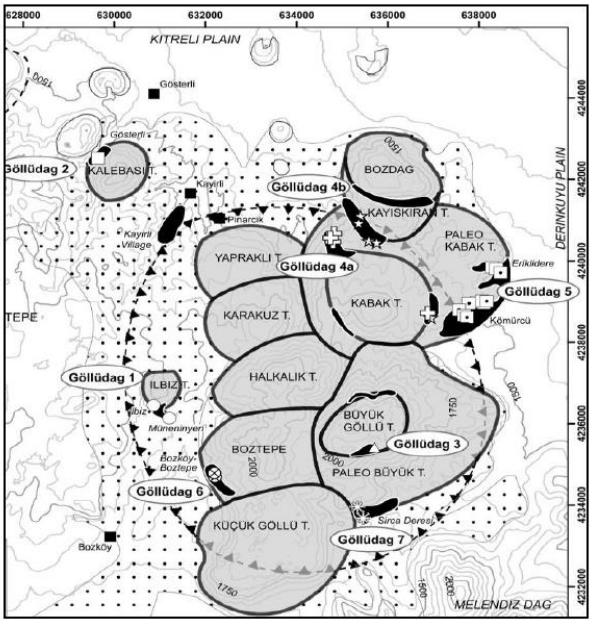


Fig. 4.1. Subagrupaciones consideradas para el volcán Göllü Dağ

(4) Este artículo introduce la división de la obsidiana según sea alcalina, peralcalina o calcalcalina. El estudio se enfoca en la obsidiana peralcalina, característica del volcán Nemrut Dağ y de una zona de Bingöl (referida como Bingöl A). Para la discusión se centra en la representación bidimensional y tridimensional de los elementos químicos de las muestras. La toma de datos se hace en mayor magnitud en el volcán Nemrut Dağ, dividendo este en diversas zonas (véase Fig. 4.2).

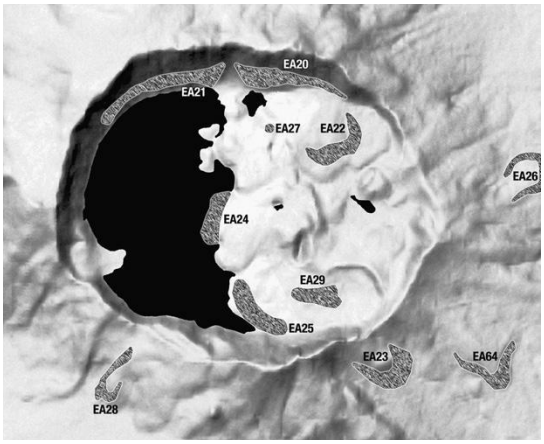


Fig. 4.2. Subagrupaciones consideradas para el volcán Nemrut Dağ

(5) Este cuarto artículo estudia la obsidiana procedente de la fuentes Bingöl A, Bingöl B y Meydan Dağ, todos de la región sudeste de Anatolia (véase la Figura 4.3). Se remarca la dificultad para la distinción entre las muestras procedentes de Bingöl A y

Nemrut Dağ. El método empleado para la caracterización de la obsidiana es la representación bidimensional de la razón entre elementos químicos.

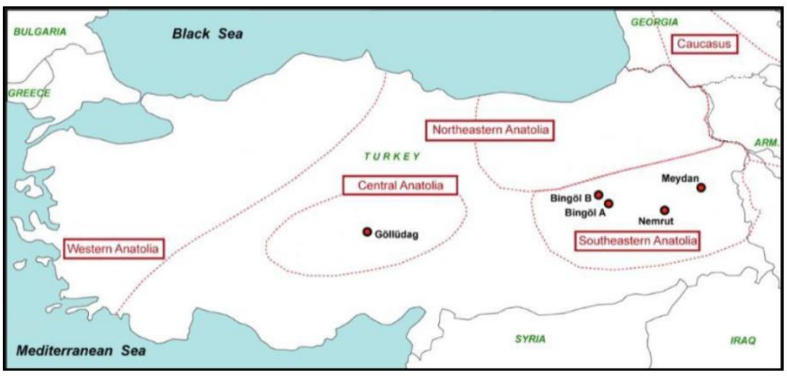


Fig. 4.3. Ubicación de los volcanes relacionados con los estudios

(6) Estudio similar a los anteriores, aunque con un mayor número de volcanes a categorizar. Utilizan dos métodos ya vistos, la representación bidimensional de concentraciones y el uso de funciones discriminantes.

(7) En este artículo se realiza por primera vez un estudio de procedencia en su totalidad. El método empleado para la recogida de artefactos consiste en situarlos en una malla del yacimiento (véase Figura 4.4). En lo referente al estudio, únicamente se realizan representaciones bidimensionales (de razones de elementos). Al encontrar una representación que agrupe bien las muestras de las fuentes de procedencia, se superponen las de los artefactos y, según si caen en una región u otra, se le asigna a uno u otro volcán.

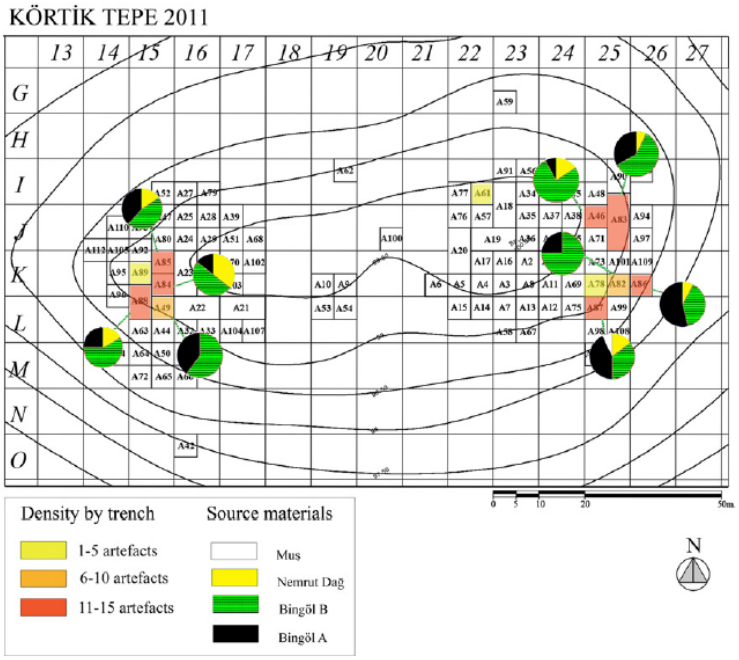


Fig. 4.4. Distribución de los artefactos de los yacimientos

(8) Realiza un análisis de procedencia en el que se busca identificar si unos artefactos determinados provienen del volcán Nenezi Dağ o del Göllü Dağ. La distinción entre los dos se realiza de forma sencilla con una representación bidimensional.

4.1.2 Estado de desarrollo

La exploración del estado de desarrollo del programa informático fue unos de los primeros puntos de estudio. El funcionamiento de éste se puede diseccionar en tres grandes grupos:

- Tratamiento de datos.
- Algoritmos de clasificación.
- Herramientas para la representación gráfica.

En cuanto al primer punto, se había realizado un **tratamiento de datos** bastante completo. Las cuatro funciones básicas implementadas fueron:

- Importar ficheros a la base de datos del programa.
- Guardar los cambios realizados en el fichero de origen.
- Exportar datos a un nuevo fichero.
- Combinar dos o más ficheros en uno sólo.

En este punto, cabe destacar que se realizó una previa recolección de los datos presentes en los artículos vistos anteriormente, quedando agrupados en un único fichero Excel con una estructura preestablecida.

En segundo lugar, como técnica de clasificación, se implementó un algoritmo de elección de mejores combinaciones basado en el **coeficiente *PseudoF***. Estas combinaciones podían ser tanto simples (elementos químicos simples) como fraccionales (razón entre dos elementos químicos). En la Figura 4.5 se observa su implementación de cara al usuario final.

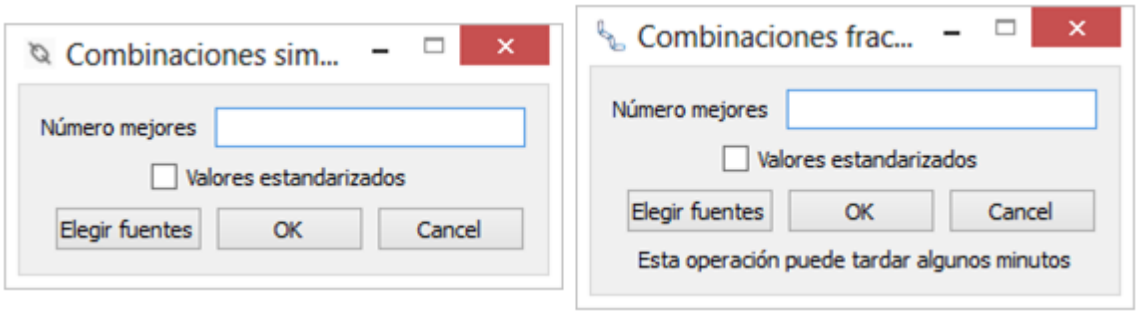


Fig. 4.5. Diálogos mejores combinaciones simples y fraccionales

En tercer lugar, se implementaron **herramientas de representación gráfica**, tanto bidimensional (Figura 4.6) como tridimensional (Figura 4.7).

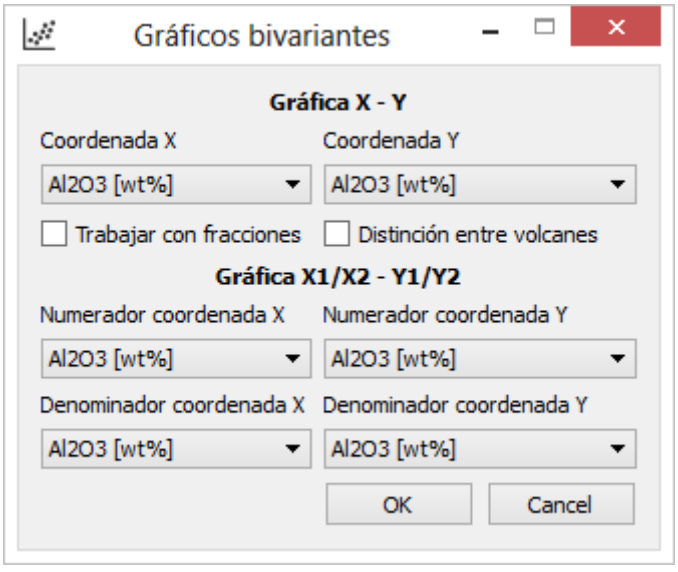


Fig. 4.6. Diálogo gráficos bidimensionales

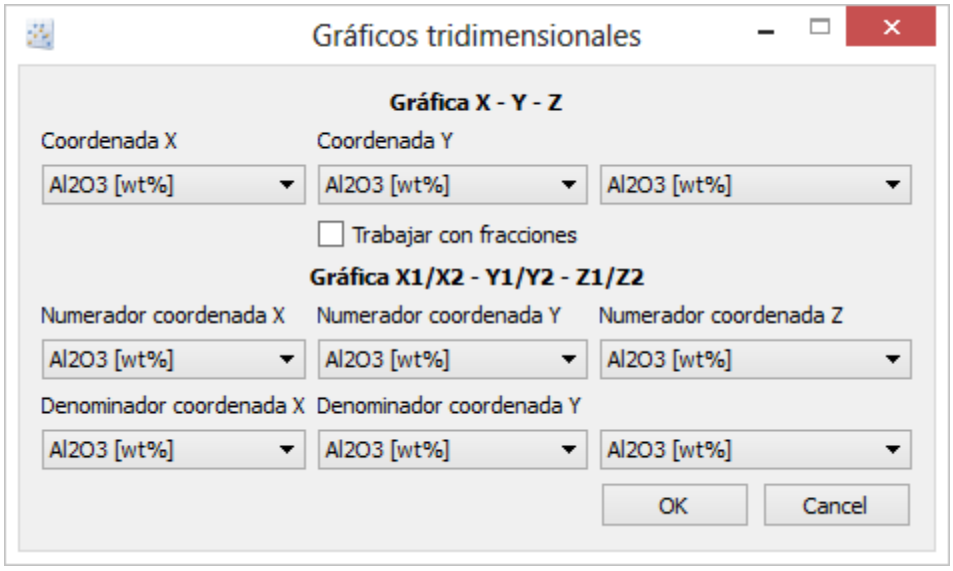


Fig. 4.7. Diálogo gráficos tridimensionales

Por último, en la Figura 4.8 se observa la interfaz principal del programa.

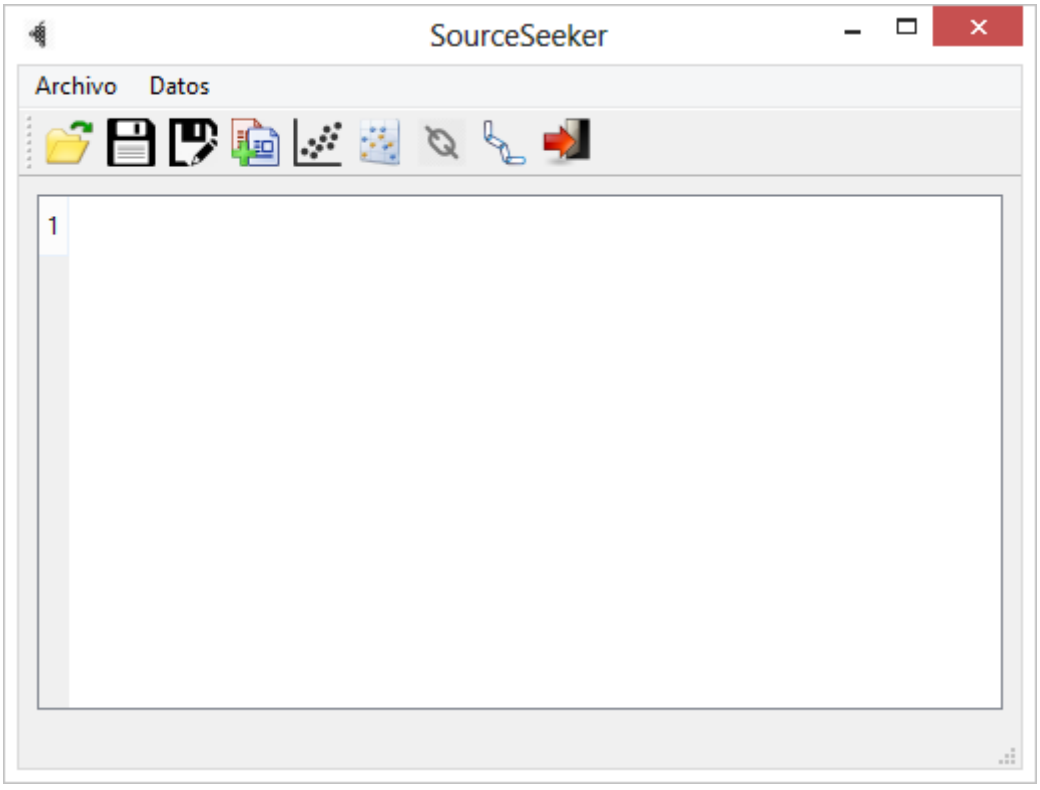


Fig. 4.8. Interfaz principal del programa

4.1.3. Vías de continuación propuestas

Entre las principales vías de continuación pueden encontrarse propuestas de carácter estadístico y de tratamiento de datos.

Propuestas estadísticas:

- *Estadística descriptiva*. Permite una descripción sintetizada de la información que se tiene con el fin de describir las características de un conjunto de datos.
- *Contraste de hipótesis*. Incluye varios procedimientos para juzgar si una propiedad que se supone en una población estadística es compatible con lo observado en la muestra de dicha población.
- *Discriminación en árbol*. Permite situar una muestra a partir del cumplimiento de una serie de condiciones representadas en forma de árbol.
- *Análisis discriminante*. Permite la clasificación de una muestra en diversos grupos conocidos de antemano en función de las propiedades que tiene.
- *Análisis de componentes principales*. Permite reducir la dimensión de un conjunto de datos perdiendo el mínimo de información.
- *Dendograma*. Aplicado a la agrupación de datos, permite ver cómo se forman las agrupaciones en función de alguna característica.

Propuesta de tratamiento de datos

En ese punto se propone la introducción de los artefactos de los yacimientos. Aunque en el trabajo anterior ya se contempló la presencia de artefactos, éstos no se diferenciaban de las fuentes. Esto provocaba que los artefactos formarán parte en los cálculos internos pudiendo influir así en los resultados finales.

Otras propuestas

De cara al usuario, se propone introducir la posibilidad de elegir las fuentes y muestras con las que se quiera trabajar en cada momento.

Por otro lado, también se propusieron algunas mejoras por parte del director del trabajo, tales como:

- *Cálculo de la elipse de error*. Equivalente al cálculo de intervalos de confianza bidimensionales.
- *Mejoras en la interfaz gráfica*. Se propuso la separación de la representación gráfica de las fuentes y de las muestras en dos tablas distintas.

4.1.4. Estudio previo de los datos

La exploración de los datos puso de manifiesto la **heterogeneidad** entre los mismos (pocos elementos comunes entre las diferentes observaciones). Este factor fue determinante para muchas de las decisiones que se tomaron posteriormente; la

mayoría de algoritmos estadísticos que se conocen no contemplan la existencia de valores perdidos. Se entiende por valor perdido aquél que no fue recogido durante el proceso de medición.

Como ya se ha comentado, el conjunto de datos del que se disponía proviene de la agrupación de diferentes informes. Éstos seguían patrones de medición distintos, factor que provocó la aparición de múltiples valores perdidos. La presencia de éstos es un problema común en los estudios de cualquier investigación e ignorarlos puede producir la aparición de sesgos.

Existen varias técnicas para lidiar con los valores perdidos. La más común y utilizada (opción por defecto en la mayoría de paquetes estadísticos) consiste en analizar sólo aquellos casos con la información completa en el conjunto de variables. Se conocen como análisis de casos completos (*Casewise*, *Listwise*). Este tipo de técnicas limitan la representatividad o validez de los resultados, ya que hay sujetos que se quedan fuera del estudio.

También encontramos técnicas de imputación de valores (imputación por la media, imputación por constante, imputación por regresión, etc.). Sin embargo, fueron descartadas debido a la gran ausencia de información que caracterizaba el conjunto de datos de estudio.

4.1.5. Conclusiones del estudio previo

Después de realizar los primeros estudios, se empezaron a descartar algunas de las vías de continuación propuestas.

Propuestas estadísticas

En primer lugar, se descartó ampliar el programa por la vía de la **estadística descriptiva**. Aunque era muy útil para realizar una primera aproximación a los datos, quedaba fuera del objetivo principal del programa.

Por otro lado, también se desechó la idea de aplicar técnicas de **contraste de hipótesis** debido al carácter multivariante del estudio.

En cuanto a los **algoritmos de agrupamiento**, se analizaron varias opciones dentro del gran abanico de técnicas posibles, entre las que destacan:

- *Dendograma* (propuesta introducida por el proyectista anterior).

- *K-means*. Algoritmo de agrupación de acuerdo a un criterio, por lo general distancia o similitud.

Se decidió decantarse por el algoritmo *k-means* debido a su popularidad y simplicidad de aplicación. Cabe decir que este algoritmo conlleva una serie de desventajas que se explicarán detalladamente más adelante (véase Algoritmos de agrupamiento).

Para el estudio de los **algoritmos de clasificación**, cabe diferenciar entre las técnicas estadísticas clásicas y las técnicas de minería de datos (*data mining*).

- *Estadística clásica*. Comprende todo tipo de técnicas como el análisis discriminante, tanto lineal (LDA) como cuadrático (QDA), y el análisis de componentes principales (PCA). Este tipo de técnicas necesitan trabajar con conjuntos de datos completos, no admitiendo valores perdidos. Por la naturaleza del juego de datos del que se requiere, quedaron fuera de los objetivos del proyecto.
- *Minería de datos*. Engloba técnicas de clasificación, tales como árboles de decisión o el algoritmo del vecino más próximo (*k-nearest neighbors*). Estos algoritmos admiten matrices de datos incompletas, adquiriendo la capacidad de trabajar con valores perdidos. Este fue el principal motivo por el que se decidió decantarse por esta vía de trabajo. Se hizo hincapié en los árboles de decisión, técnicas muy representativas y visuales que combaten con relativa solvencia conjuntos de datos heterogéneos.

Por último, resultaba interesante introducir técnicas de **validación de modelos**. Uno de los métodos de validación más utilizados en la mayoría de paquetes estadísticos es el algoritmo conocido como *cross validation*. Su implementación permite cuantificar el grado de certeza de la predicción que genera nuestro modelo.

Propuestas de tratamiento de datos

La introducción de los artefactos de los yacimientos era uno de los puntos básicos de mejora. Su aplicación era necesaria tanto para evitar influencias en los cálculos internos como para la implementación de las nuevas técnicas de minería de datos.

Otras propuestas

Respecto a las propuestas de carácter gráfico se planteó una redistribución de la interfaz principal del programa que contemplara todas las propuestas realizadas en ese aspecto.

Por último, una de las propuestas en la que más hincapié se puso fue la implementación del cálculo de la elipse de error.

Se completaba de este modo la lista total de mejoras a cubrir:

- Introducción de los artefactos de los yacimientos para permitir la separación de fuentes y muestras.
- Redistribución de la interfaz gráfica de usuario.
- Algoritmo *k-means* como técnica de agrupamiento.
- Árboles de decisión como algoritmo de clasificación.
- *Cross validation* como algoritmo de validación.
- Cálculo de la elipse de error.

4.2. Primer enfoque

Una vez efectuada la lista de mejoras a implementar, se realizó una primera aproximación a las posibles soluciones.

Algunas de las mejoras implicaban cambios estructurales en el código que se había escrito hasta el momento. La introducción de los artefactos de los yacimientos suponía un replanteamiento del tratamiento de datos, implicando una reestructuración en la forma de almacenarlos y procesarlos.

Por otro lado, se estudiaron las diversas posibilidades existentes para aplicar las técnicas estadísticas propuestas. Para ello, se analizaron las ventajas y desventajas de los principales **paquetes estadísticos** disponibles para Python (véase Tabla 4.1).

	Scikit-learn	Statsmodels	Mlpy	MDP	MILK	pyMVPA	Orange
Análisis de componentes principales (*PCA*)	✓		✓	✓			✓
Análisis discriminante lineal (*LDA*)	✓		✓	✓		✓	
Análisis	✓			✓			

discriminante cuadrático (*QDA*)							
ANOVA		✓				✓	
Árbol de decisiones	✓		✓	✓			✓
Cross validation	✓				✓		✓
Dendograma (*clustering*)	✓		✓	✓			✓
K-means (*clustering*)	✓		✓		✓		✓
Random Forest	✓			✓	✓		✓

Tabla 4.1. Comparación de los principales paquetes estadísticos para Python

Los dos paquetes más adecuados para el desarrollo del trabajo eran *scikit-learn* y *orange*. Ambos implementaban la mayoría de algoritmos que se querían aplicar a nuestro programa.

Scikit-learn es uno de los paquetes estadísticos más conocidos y utilizados en Python. Presenta una gran variedad de algoritmos de agrupación, clasificación, imputación, etc. Además presenta una documentación muy extensa. Al ser muy popular, puede encontrarse todo tipo de información y tutoriales acerca de su utilización.

La gran desventaja de este paquete reside en el factor de que *Scikit-learn* no puede trabajar con valores perdidos. Aunque contempla métodos de imputación de valores, requiere del uso de matrices completas para todas sus técnicas implementadas. Por ello, se descartó su uso.

Orange es una librería de Python orientada a la minería de datos. Aunque cuenta con una documentación extensa (9) sobre su utilización, no es un paquete muy popular. Resulta difícil encontrar otro tipo de información o tutoriales fuera del ámbito de la documentación oficial.

Una de las grandes ventajas de este paquete estadístico y motivo de su elección es la posibilidad de trabajar, además de contemplar métodos de imputación, con valores perdidos.

4.3. Tratamiento de datos

Como ya se ha comentado, era necesario un cambio en el tratamiento de datos, del tal forma que se pudieran diferenciar las muestras de las fuentes.

A tal fin, fue creada una clase base llamada 'Medicion'. El concepto principal de esta clase era el de almacenar el conjunto de valores de una medición determinada para poder acceder a ellos de forma rápida y sencilla. También el de comprobar si una determinada medición contenía el valor de un determinado compuesto.

Posteriormente, se crearon dos clases llamadas 'Muestra' y 'Fuente'. Estas clases heredaron de la clase 'Medicion', añadiendo algunas funcionalidades especificas (véase Figura 4.9).

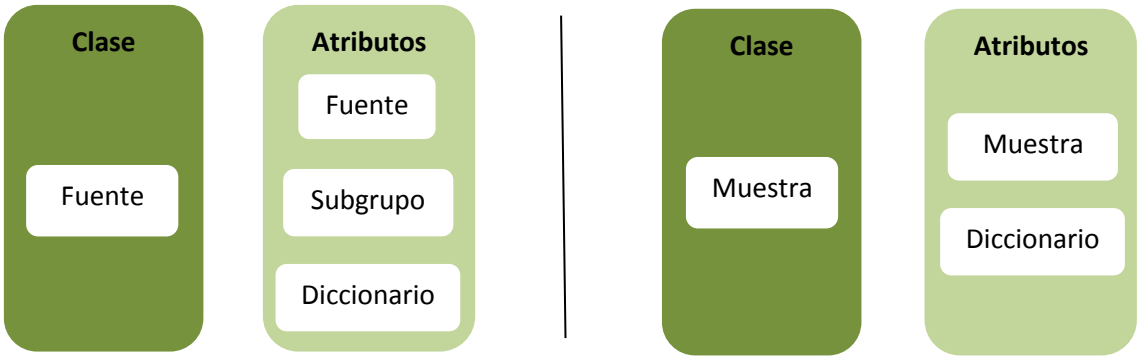


Fig. 4.9. Estructura de la clase 'Fuente' y 'Muestra'

Por último, se creó un objeto de almacenaje para cada tipo de medición, implementados como las clases 'DataFuente' y 'DataMuestra'. Estas clases fueron concebidas con la idea de poder seleccionar y extraer la información deseada en cada momento, como también de hacerlo de forma afín al paquete estadístico elegido.

Todos estos cambios supusieron modificar la forma en que el usuario final interactuaba con el conjunto de datos. Se mantuvieron las funciones básicas presentes del programa anterior, sin embargo, éstas fueron duplicadas (una para cada tipo de medición). De esta forma, el usuario podría elegir sobre qué conjunto de datos actuar en cada momento (véase Figura 4.10).

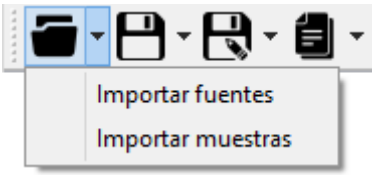


Fig. 4.10. Nueva interfaz para la importación de datos

El hecho de separar las muestras de las fuentes permitió tener en ficheros diferentes los datos referentes a las fuentes y los datos referentes a las muestras. Esto conllevó una reestructuración en la plantilla de datos que se seguía hasta el momento (véase Tablas 4.2 y 4.3).

Fuente	Subgrupo	Elemento 1 [unidades]	Elemento 2 [unidades]	...	Elemento k [unidades]
f1	g1	valor 1.1	valor 1.2	...	valor 1.k
...	...	...	...	...	...
fn	gn	valor n.1	valor n.2	...	valor n.k

Tabla 4.2. Esquema de la estructura de las fuentes de Excel

Muestra	Elemento 1 [unidades]	Elemento 2 [unidades]	...	Elemento k [unidades]
m1	valor 1.1	valor 1.2	...	valor 1.k
...	...	...	...	...
mn	valor n.1	valor n.2	...	valor n.k

Tabla 4.3. Esquema de la estructura de las muestras de Excel

Enfatizar que el uso del paquete estadístico *Orange* supuso una incompatibilidad a la hora de utilizar acentos u otros caracteres especiales. Esto conllevó una reconversión de los nombres de las fuentes, subgrupos o compuestos que contenían alguno de estos caracteres.

4.4. Mejoras en la interfaz gráfica

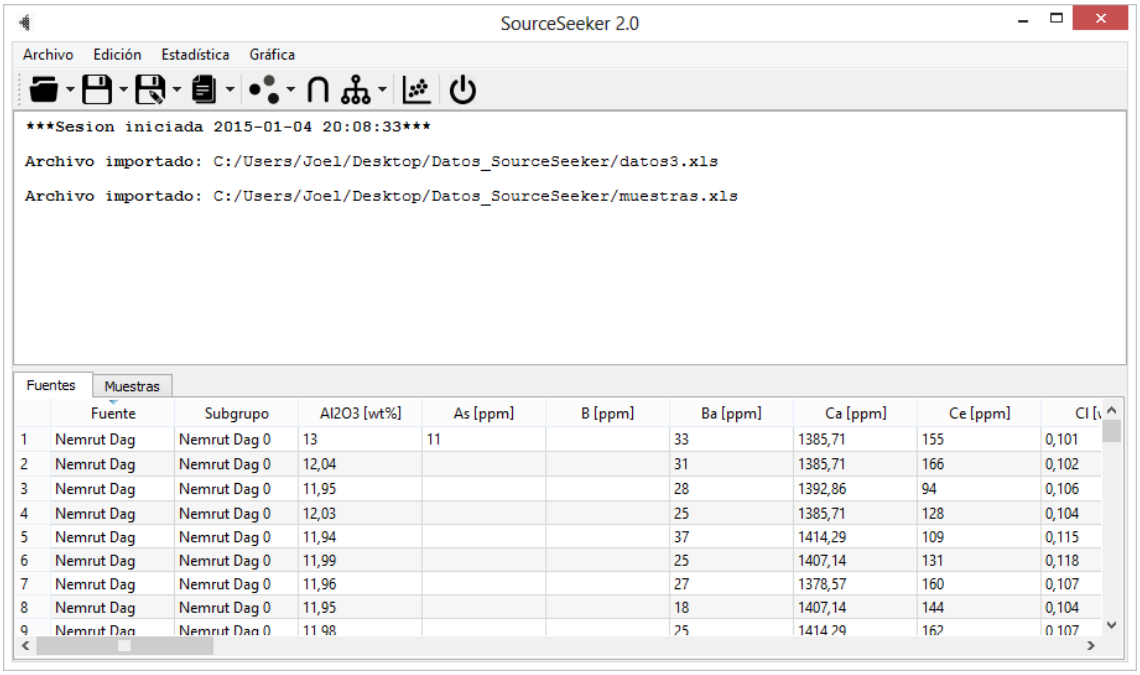
Tras modificar el tratamiento de datos se procedió a la reestructuración de la interfaz gráfica de usuario, de tal forma que:

- Se pudiese visualizar de forma rápida y sencilla los resultados de los diferentes algoritmos que se aplicasen.
- Se pudiese separar de forma gráfica la representación de datos referentes a fuentes o muestras.

Para lograr la primera de estas premisas, se añadió un *widget* (*QTextEdit*) a la estructura principal de la interfaz. Este *widget* se caracteriza por ser compatible con código HTML, por lo que la introducción de texto con formato, listas o tablas se podía realizar de manera rápida y sencilla. La compatibilidad con la funcionalidad de copiado (*Ctrl+V*) permite el traslado de la información que contiene a cualquier otro documento. Cabe destacar que se deshabilitó la opción de poder introducir contenido a través del teclado (*widget* de sólo lectura). Durante el resto del trabajo se hará referencia a este *widget* como **hoja de sesión**.

En cuanto a la segunda premisa, se añadió un *widget* (*QTabWidget*) donde poder albergar una tabla referente a las fuentes y otra referente a las muestras. Remarcar que se ha modificado el tipo de *widget* que contienen los datos en la interfaz, pasando de utilizar un *QTableWidget* a un *QTableView*. Así, al cambiar cualquier elemento de la tabla, automáticamente queda modificado en el modelo interno sin tener que guardarlo. En contrapartida, se han perdido algunas de las funcionalidades respecto al programa anterior, como la funcionalidad de ordenado y copiado-pegado. Sin embargo, éstas pueden ser implementadas de nuevo en versiones futuras.

En la Figura 4.11 se observa el resultado final de la interfaz gráfica.



	Fuente	Subgrupo	Al2O3 [wt%]	As [ppm]	B [ppm]	Ba [ppm]	Ca [ppm]	Ce [ppm]	Cl [
1	Nemrut Dag	Nemrut Dag 0	13	11		33	1385,71	155	0,101
2	Nemrut Dag	Nemrut Dag 0	12,04			31	1385,71	166	0,102
3	Nemrut Dag	Nemrut Dag 0	11,95			28	1392,86	94	0,106
4	Nemrut Dag	Nemrut Dag 0	12,03			25	1385,71	128	0,104
5	Nemrut Dag	Nemrut Dag 0	11,94			37	1414,29	109	0,115
6	Nemrut Dag	Nemrut Dag 0	11,99			25	1407,14	131	0,118
7	Nemrut Dag	Nemrut Dag 0	11,96			27	1378,57	160	0,107
8	Nemrut Dag	Nemrut Dag 0	11,95			18	1407,14	144	0,104
9	Nemrut Dag	Nemrut Dag 0	11,98			25	1414,29	162	0,107

Fig. 4.11. Interfaz principal del programa

4.5. Algoritmos de agrupamiento

Los algoritmos de agrupamiento (*clustering*) son procedimientos de agrupación de acuerdo a un criterio, por lo general distancia o similitud. *Orange* ofrece tres algoritmos de agrupamiento distintos:

- K-means clustering (`kmeans`)
- Hierarchichal clustering (`hierarchical`)
- Consensus clustering (`consensus`)

Como ya se ha dicho, en el presente trabajo se implementa la **técnica *k-means*** dada su gran popularidad y simplicidad de aplicación. La mayor desventaja de esta técnica radica en el hecho de tener que indicar una serie de parámetros iniciales condicionantes del resultado final.

4.5.1. Algoritmo *k-means*

Este algoritmo trata de dividir un conjunto de *N* observaciones en un cierto número *K* de conglomerados (*clusters*) fijados previamente. El algoritmo *k-means* se implementa:

1. Elección del número de conglomerados k.
2. Elección de un conjunto de k centroides iniciales.

3. Asignación de cada observación del conjunto de datos al centroide más cercano.
4. Cálculo, para cada conglomerado, del nuevo centroide como baricentro de las agrupaciones resultantes en el paso anterior.
5. Volver al paso tres hasta que el criterio de convergencia es alcanzado (ninguna observación se mueve de grupo).

En la Figura 4.12 se observa un diagrama de flujo del algoritmo presentado.

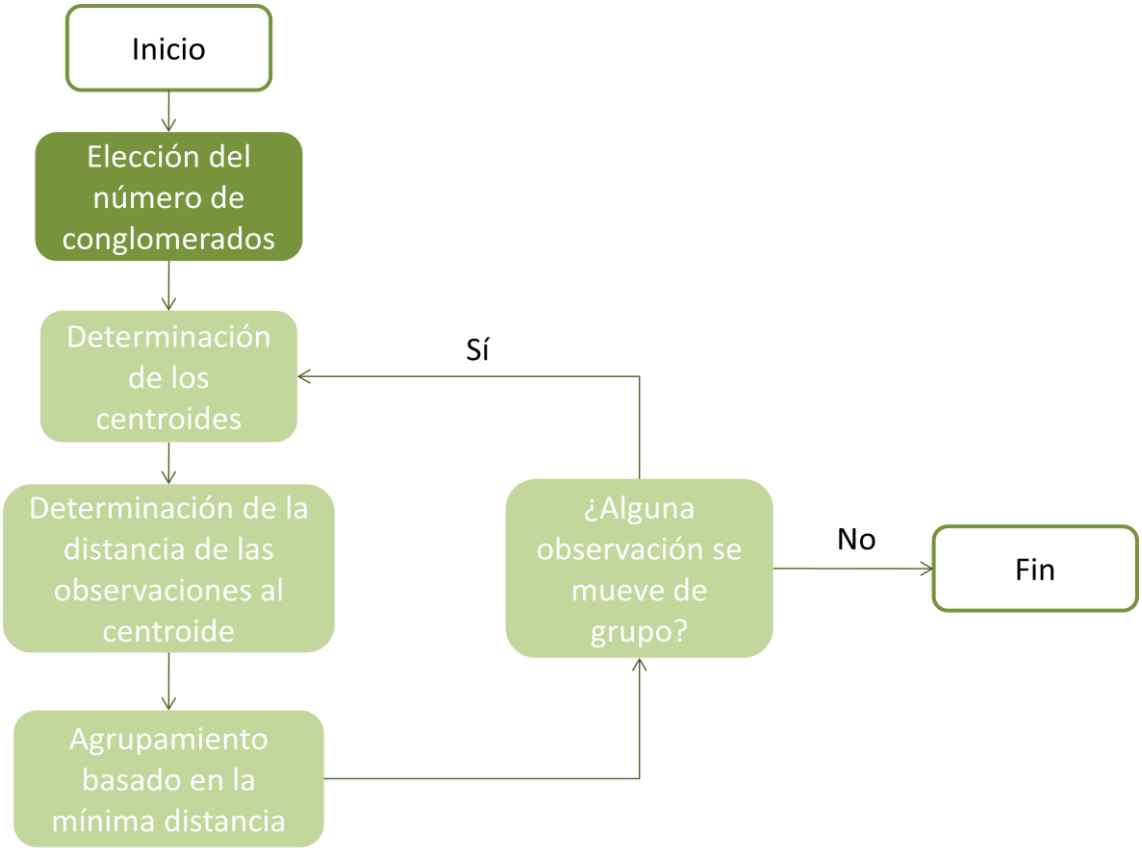


Fig. 4.12. Diagrama de flujo *k-means*

4.5.1.1. Elección del número de conglomerados.

Este algoritmo necesita la especificación, como parámetro inicial, del número de conglomerados por el que está formado el conjunto de datos a estudiar. Existen varias técnicas que facilitan la decisión, entre las que destaca el **coeficiente *silhouette***.

Este coeficiente asigna una serie de valores a un rango de conglomerados determinado, siendo el conglomerado con un coeficiente más elevado el más adecuado. Está definido para cada observación y está compuesto por dos valores:

- **a:** Distancia media entre una observación y el resto de observaciones que pertenecen al mismo conglomerado.
- **b:** Distancia media entre una observación y el resto de observaciones en el siguiente conglomerado más cercano a la observación.

El coeficiente *silhouette s* para una observación viene dado por la siguiente expresión:

$$s = \frac{b - a}{\max(a, b)}$$

Para un conjunto de observaciones este coeficiente viene dado por la media de los coeficientes *silhouette* de cada una de las observaciones.

Cabe remarcar que se trata de una técnica orientativa, por lo que se deben hacer exploraciones previas de los datos para llegar a las soluciones más óptimas posibles.

4.5.1.2. Elección de centroides iniciales

La elección de centroides iniciales es uno de los parámetros más importantes para el correcto funcionamiento del algoritmo. *Orange* ofrece varias técnicas de inicialización, entre las que destacan:

- *init_random:*

Se basa en una elección pseudoaleatoria de los centroides iniciales. La mayor parte de los generadores de números aleatorios son, en realidad, pseudoaleatorios; se calcula (o introduce internamente) un valor x_0 llamado semilla y, a partir de él, se generan el resto de valores. De este modo, siempre que se parta de la misma semilla, se obtendrá la misma secuencia de valores y, por tanto, cada vez que se ejecute el algoritmo se obtendrán los mismos resultados.

- *init_diversity:*

Devuelve un conjunto de centroides donde el primero de ellos está lo más alejado posible del centro de los datos. Los centroides siguientes se calculan con la premisa de que la distancia mínima a los centroides calculados anteriormente sea máxima.

4.5.1.3. Criterios de asignación a un conglomerado.

Orange ofrece un gran abanico de distancias como criterios de asignación de una observación a un conglomerado:

- *Hamming:*

Número de variables en que difieren dos observaciones. No es apropiada para observaciones que contienen variables continuas.

- *Maximal:*

La distancia entre dos observaciones viene determinada por la máxima diferencia de una determinada variable.

- *Manhattan:*

Suma de las diferencias (absolutas).

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- *Eucledian:*

Distancia 'ordinaria' entre dos puntos. Se deduce a partir del teorema de Pitágoras:

$$d(p,q) = \sum_{i=1}^{n} (p_i - q_i)^2$$

- *Relief:*

Similar a la distancia Manhattan pero incorporando el tratamiento de valores indefinidos.

- *PearsonR:*

Disimilaridad de *Pearson:*

$$d = \frac{1-r}{2}$$

Dónde *r* es el coeficiente de *Pearson* (10)

- *SpearmanR:*

Disimilaridad de *Spearman*:

$$d = \frac{1-r}{2}$$

Dónde r es el coeficiente de *Spearman*. (11)

- *Mahalanobis:*

A diferencia de la distancia *eucledian*, tiene en cuenta la correlación entre las variables aleatorias.

4.5.2. Procesamiento previo de los datos

Como se ha comentado con anterioridad, los datos son poco homogéneos. Sin embargo, si únicamente se analiza una fuente hay bastantes compuestos comunes para todas las observaciones.

Al aplicar el algoritmo *k-means* o el cálculo del coeficiente *silhouette* se ha implementado la posibilidad de eliminar aquellos compuestos que no explican todas las observaciones. De este modo, es posible mejorar los resultados a obtener.

4.5.3. Implementación del coeficiente *silhouette*

Para la adición al programa del cálculo del coeficiente *silhouette* se creó un diálogo en el que podían modificarse los siguientes parámetros:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Rango de conglomerados.* Permite seleccionar el rango de conglomerados en el que se quiere trabajar.
- *Semilla.* Permite seleccionar el valor de la semilla.
- *Inicialización.* Permite seleccionar el método de inicialización de los centroides.
- *Distancia.* Permite seleccionar el tipo de distancia con la que se quiere trabajar.
- *Graficar resultados.* Posibilidad de graficar los resultados de salida.
- *Eliminar compuestos que no explican todas las muestras.* Todo aquel compuesto para el que alguna de las observaciones no contenga su valor será ignorado en el cálculo.

En la Figura 4.13 se observa el aspecto final de dicho diálogo.

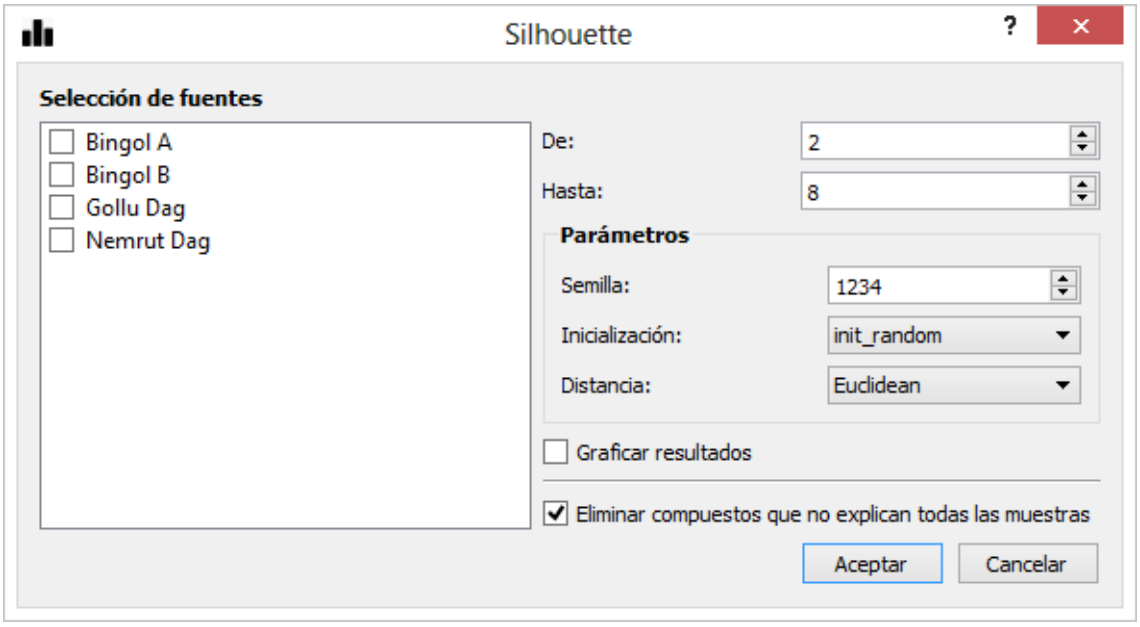


Fig. 4.13. Dialogo *silhouette*

Al ejecutar el algoritmo aparece en la hoja de sesión una tabla en la que se muestran los coeficientes *silhouette* para cada número de conglomerados dentro del rango seleccionado (véase Figura 4.14).

Fuente	2	3	4	5	6	7	8
Bingol B	0.031	0.119	0.452	0.803	1.000	0.000	0.000
Bingol A	0.551	0.649	0.642	0.506	0.551	0.389	0.401
Nemrut Dag	0.620	0.488	0.407	0.413	0.384	0.446	0.409
Gollu Dag	0.433	0.299	0.413	0.355	0.416	0.375	0.318

Fig. 4.14. Tabla de resultados para el coeficiente *silhouette*

Como se ha dicho anteriormente, la salida de resultados puede ser también en forma de gráfico (véase Figura 4.15).

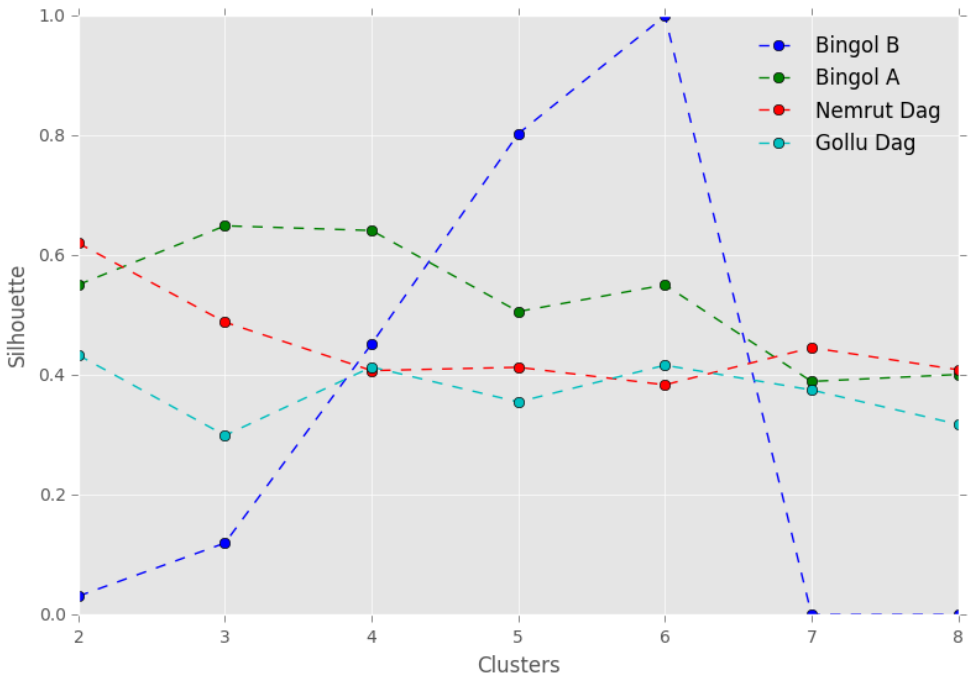


Fig. 4.15. Gráfico de resultados para el coeficiente *silhouette*

La interpretación de los resultados es simple. Para una fuente determinada, el número de conglomerados que contiene el mayor coeficiente *silhouette* es el número de conglomerados más adecuado (p.ej. para la fuente Nemrut Dağ el número de conglomerados recomendado es 2).

4.5.4. Implementación algoritmo *k-means*

La implementación de la técnica *k-means* se llevó a cabo mediante la creación de un diálogo con varios parámetros a modificar (véase Figura 4.16):

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Número clusters.* Permite seleccionar el número de conglomerados.
- *Semilla.* Permite seleccionar el valor de la semilla.
- *Inicialización.* Permite seleccionar el método de inicialización de los centroides.
- *Distancia.* Permite seleccionar el tipo de distancia con la que se quiere trabajar.
- *Eliminar compuestos que no explican todas las muestras.*

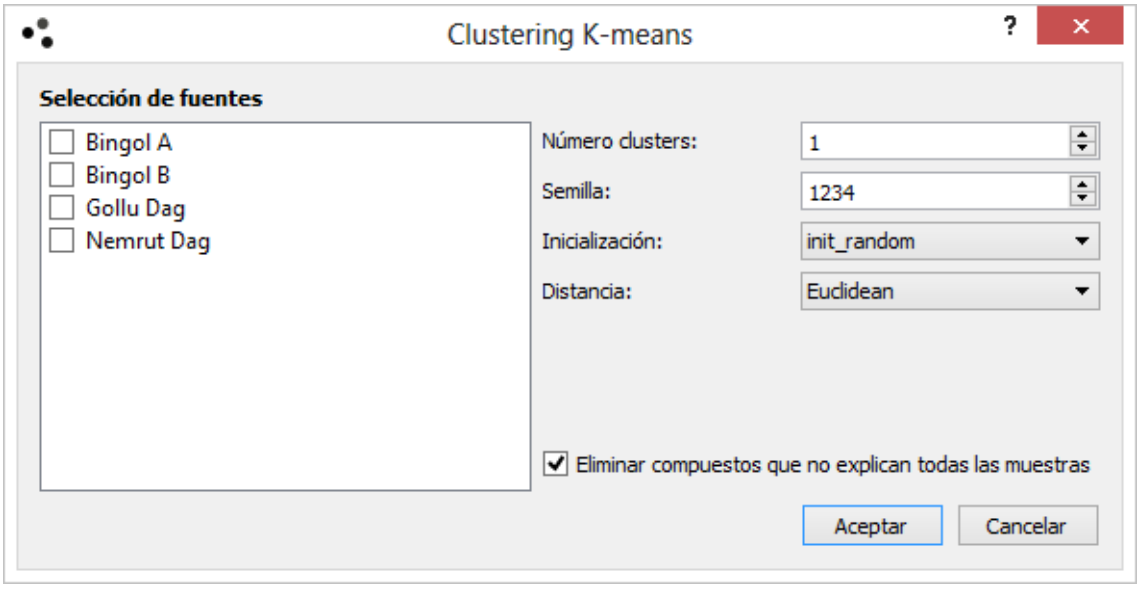


Fig. 4.16. Diálogo algoritmo *k-means*

La ejecución del algoritmo conlleva la modificación automática del atributo *subgrupo* de cada una de las observaciones de las fuentes previamente elegidas. El nombre de cada subgrupo está formado por el nombre de la fuente principal seguido de un número entero que diferencia los diversos subgrupos.

En la Figura 4.17 y 4.18 se observa un ejemplo de aplicación del algoritmo.

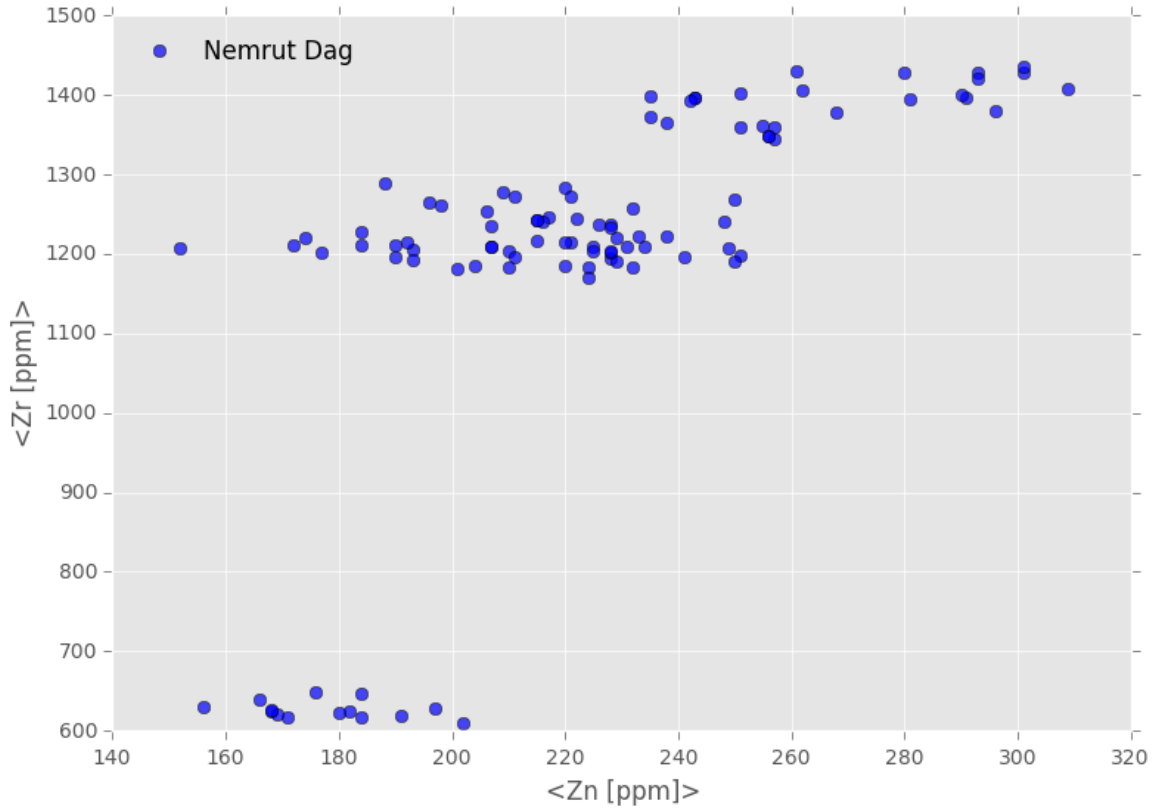


Fig. 4.17. Representación Zn [ppm] - Zr [ppm]. Anterior a la aplicación de *k-means*

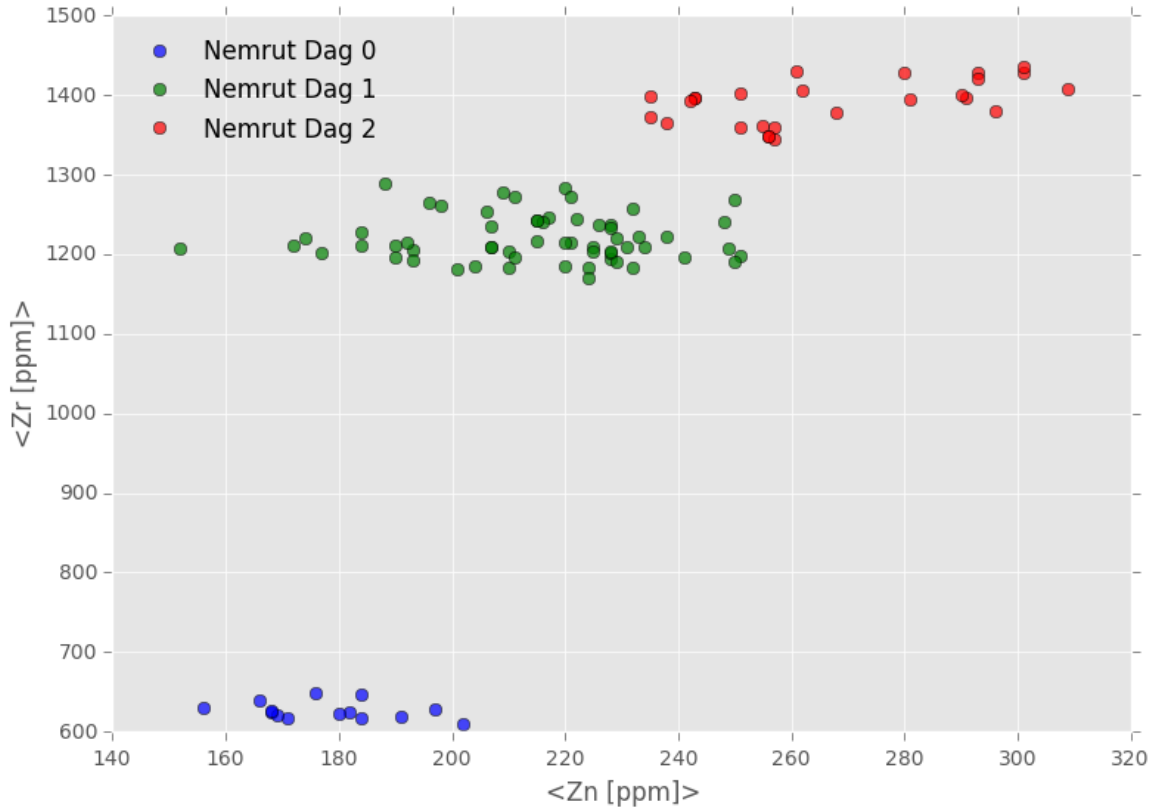


Fig. 4.18. Representación Zn [ppm] - Zr [ppm]. Posterior a la aplicación de *k-means*

4.6. Mejores combinaciones

La técnica de mejores combinaciones se implementó en el proyecto anterior. Sin embargo, ha sufrido algunas actualizaciones:

- Reagrupamiento de los diálogos (combinaciones simples y fraccionales) en uno sólo.
- Exclusión de las muestras en los cálculos internos.

El nuevo diálogo cuenta con los siguientes parámetros a modificar (véase Fig. 4.19):

- *Fuentes obligadas.* Permite seleccionar las fuentes que obligatoriamente entran en juego en el algoritmo.
- *Número mejores.* Permite seleccionar el número de mejores combinaciones que se muestran.
- *Tipo.* Permite seleccionar el tipo de combinaciones con el que se quiere trabajar. Puede ser tanto simple como fraccional.
- *Valores estandarizados.* Permite estandarizar los valores.
- *Graficar resultados.* Permite visualizar en forma de gráfico bidimensional, las mejores combinaciones de salida.

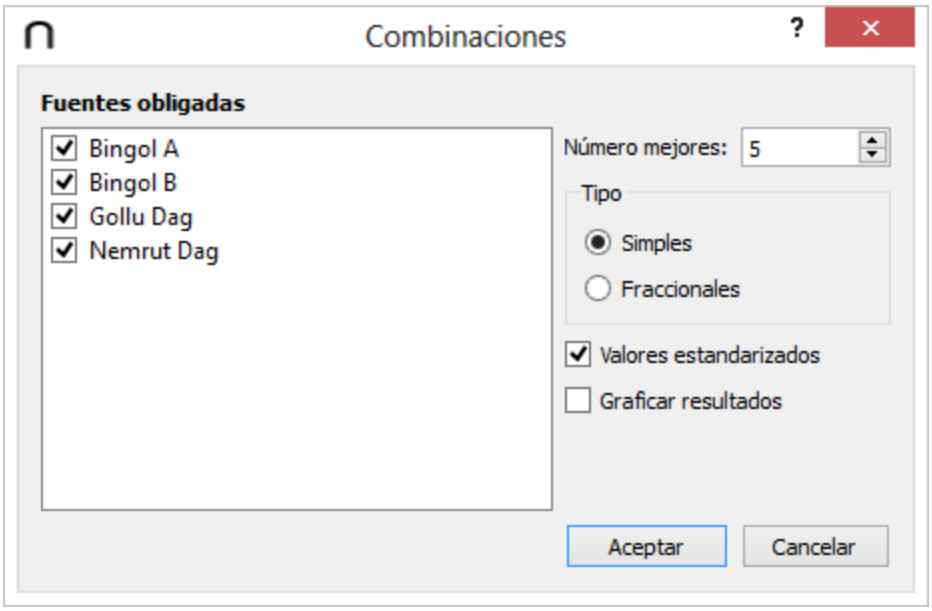


Fig. 4.19. Diálogo mejores combinaciones

Al ejecutar el algoritmo se muestra en la hoja de sesión una tabla con los resultados finales (véase Figura 4.20).

Compuesto	Compuesto	Indice
Zn [ppm]	Zr [ppm]	276.396200849
Ca [ppm]	Zr [ppm]	146.428502817
Ba [ppm]	Ca [ppm]	126.70147687
Fe [ppm]	Zr [ppm]	124.370744734
Ba [ppm]	Zr [ppm]	117.384831245

Fig. 4.20. Resultados algoritmo mejores combinaciones

4.7. Algoritmos de clasificación

Orange implementa varios algoritmos de clasificación. Entre ellos destacan:

- *Naive Bayes classifier* (`bayes`)
- *k-nearest neighbors* (`knn`)
- *Support Vector Machines* (`svm`)
- *Classification trees* (`tree`)

El paquete *Orange* ofrece también la posibilidad de implementar *ensembles.* Los algoritmos *ensemble* utilizan múltiples modelos de clasificación para mejorar el rendimiento de predicción. *Orange* implementa los siguientes:

- *Boosting*
- *Bagging*
- *Stacking*
- *Random Forest*

4.7.1. Algoritmo árboles de decisión

Los árboles de decisión son una técnica de minería de datos que crean un modelo de clasificación basado en diagramas de flujo. Describe una estructura en la que cada hoja representa las clasificaciones y cada nodo representa una conjunción de características que conducen a estas clasificaciones.

Algunas ventajas de los árboles de decisión son:

- Simples de entender e interpretar.
- Requieren una mínima preparación previa de los datos. Admite valores perdidos.

- Permite trabajar tanto con variables continuas como con variables discretas.

Las desventajas de los árboles de decisión son:

- Se pueden generar modelos sobreajustados que no generalizan bien el conjunto de datos. Este efecto se conoce como *overfitting.*
- Pueden ser inestables. Pequeñas variaciones en los datos pueden generar árboles completamente diferentes.
- Se pueden generar árboles de decisión sesgados si alguna de las fuentes domina.

Los árboles de decisión son altamente configurables y adaptables a todo tipo de datos. Entre los principales parámetros a configurar encontramos:

- *Splitter.* Representa el criterio de división utilizado en la construcción del árbol.
- *Descender.* Representa el criterio en que una muestra u observación es clasificada (desciende) a través del árbol. No influye en el proceso de construcción.
- *Máxima profundidad.* Representa el número máximo de ramas permitidas. Útil cuando la dimensión del árbol es muy grande y resulta casi ininteligible.

4.7.1.1. Elección *splitter*

Orange implementa varios criterios de división a los que se pueden acceder a través del parámetro *splitter*:

- *Splitter_IgnoreUnknowns:*

Ignora las observaciones para las que no puede ser determinada una rama individual.

- *Splitter_UnknownsToCommon:*

Coloca todos los casos ambiguos en la rama con el mayor número de casos. En el caso de que existan múltiples ramas, se selecciona una al azar y ésta es utilizada para el resto de observaciones.

- *Splitter_UnknownsToAll:*

Divide los casos con un valor desconocido de la característica en todas las ramas posibles.

- *Splitter_UnknownsToRandom:*

Selecciona una rama aleatoria para casos ambiguos.

- *Splitter_UnknownsToBranch:*

Construye una rama adicional para casos ambiguos. La descripción de la rama es '*unknown*'

- *Splitter_UnknownsAsBranchSizes:*

Divide los casos con valor desconocido en función de la proporción de casos en cada rama.

- *Splitter_UnknownsAsSelector:*

Divide los casos con valor desconocido en función de la distribución propuesta por el selector de ramas (normalmente la proporción de casos en cada rama).

4.7.1.2. Elección *descender*

El parámetro *descender* acepta las siguientes configuraciones:

- *Descender_UnknownToBranch:*

Clasifica los casos con valor desconocido en una rama especial. Sólo tiene sentido si el árbol se construye con el *Splitter_UnknownsToBranch.*

- *Descender_UnknownToCommonBranch:*

Clasifica los casos con valores desconocidos en la rama con el mayor número de casos. En el caso de que existan múltiples ramas, se selecciona al azar una de ellas para cada observación.

- *Descender_UnknownToCommonSelector:*

Clasifica los casos con valores desconocidos en la rama que adquiere la máxima recomendación por el selector.

- *Descender_UnknownMergeAsBranchSizes:*

Los subárboles (árbol que crece a partir del nodo en que se encuentra la observación) votan por la clasificación de la observación; el voto se pondera de acuerdo con los tamaños de las ramas.

- *Descender_UnknownMergeAsSelector:*

Los subárboles votan por la clasificación de la observación; el voto se pondera de acuerdo con las propuestas de los selectores.

4.7.2. *Random Forest*

Para combatir los posibles problemas introducidos por los árboles de decisión (árboles sobreajustados, un solo compuesto decide la clasificación de la muestra, dominio de una fuente, etc.) fue implementada la técnica *Random Forest.* Este algoritmo consiste en una combinación de árboles de decisión, dependiendo cada uno de ellos de una serie de variables aleatorias. La predicción final viene dada por la moda (valor que más se repite) de las predicciones de cada uno de los árboles individualmente. Además, esta técnica implementaba la posibilidad de calcular la importancia que adquiría cada una de las variables del conjunto de datos en el proceso de predicción, factor muy interesante para la caracterización de fuentes.

4.7.3. Preparación previa de los datos

Aunque los árboles de decisión permiten trabajar con valores perdidos, era necesario realizar una preparación previa de los datos para aumentar la robustez del algoritmo y evitar el cálculo de árboles sesgados.

Por ello, pareció interesante añadir la opción de que el usuario pudiera eliminar aquellos compuestos que no explicaban las muestras y/o eliminar aquellos compuestos que no explicaban algunas de las fuentes seleccionadas.

4.7.4. Implementación árbol de decisiones

Para la adición al programa de los árboles de decisión se creó un diálogo en el que se podían modificar varios parámetros. Por una parte se encuentran:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Selección de muestras.* Permite seleccionar las muestras con las que se quiere trabajar.
- *Splitter*. Permite seleccionar el tipo de *splitter* deseado.
- *Descender.* Permite seleccionar el tipo de *descender* deseado.
- *Profundidad.* Permite limitar la profundidad máxima del árbol resultante.
- *Diferenciar entre subgrupos dentro de una misma fuente.*
- *Mostrar árbol de decisión.*
- *Cross validation.* Técnica de validación del modelo.

Por otra parte, admite indicar el procesamiento de datos deseado.

- Eliminar compuestos que no contiene la muestra.[1]
- Eliminar compuestos que no explican la fuente.

En la Figura 4.21 se observa el resultado final del diálogo.

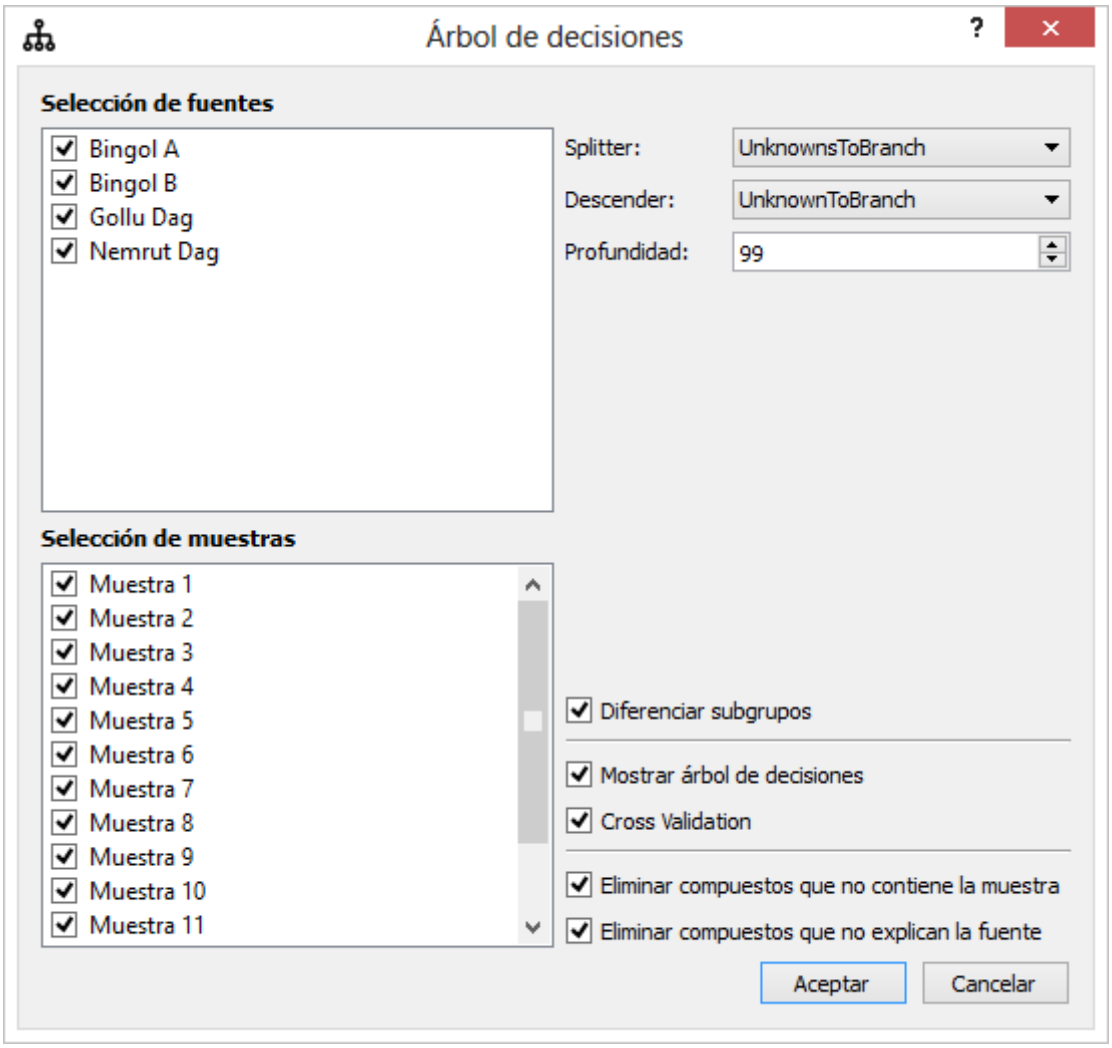


Fig. 4.21. Diálogo árboles de decisión

En la Figura 4.22 se observa un ejemplo de árbol de decisión. El resultado es un diagrama de flujo muy manejable e inteligible para el usuario final. El porcentaje final de cada rama indica el porcentaje de observaciones de la predicción de la rama respecto al total de observaciones. Puede apreciarse que algunas de las ramas no se han desarrollado hasta el final (se considera que una rama termina cuando el porcentaje es del 100%), por lo que la limitación de profundidad ha actuado en esos casos.

[1] No se recomienda realizar análisis incluyendo muestras que tengan compuestos muy dispares entre ellas. Esto reduce significativamente el número total de variables con las que trabaja el algoritmo.

```
Ba [ppm]>467.500: Gollu Dag 0 (100.00%)
Ba [ppm]<=467.500
|     Zn [ppm]<=53.500: Bingol B 0 (100.00%)
|     Zn [ppm]>53.500
|     |     Zr [ppm]<=491.000: Bingol B 0 (100.00%)
|     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Zr [ppm]>491.000
|     |     |     Zr [ppm]<=909.500: Nemrut Dag 0 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
|     |     |     Zr [ppm]>909.500
|     |     |     |     Zr [ppm]<=1319.000: Nemrut Dag 1 (89.47%)
|     |     |     |     Zr [ppm]>1319.000: Bingol A 0 (52.00%)
|     |     |     |     Zr [ppm]=unknown<null node>: <null node>
|     Zn [ppm]=unknown
|     |     Ba [ppm]<=35.000: Bingol A 0 (100.00%)
|     |     Ba [ppm]>35.000: Gollu Dag 0 (100.00%)
|     |     Ba [ppm]=unknown<null node>: <null node>
Ba [ppm]=unknown
|     Zn [ppm]<=108.000: Gollu Dag 0 (100.00%)
|     Zn [ppm]=unknown<null node>: <null node>
|     Zn [ppm]>108.000
|     |     Zr [ppm]<=1308.000: Nemrut Dag 1 (100.00%)
|     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Zr [ppm]>1308.000
|     |     |     Zr [ppm]<=1397.000: Nemrut Dag 2 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
|     |     |     Zr [ppm]>1397.000
|     |     |     |     Zn [ppm]<=254.000: Nemrut Dag 2 (100.00%)
|     |     |     |     Zn [ppm]>254.000: Bingol A 0 (50.00%)
|     |     |     |     Zn [ppm]=unknown<null node>: <null node>
```

Fig. 4.22. Ejemplo árbol de decisión

En la Figura 4.23 se observa, a modo de ejemplo, el resultado de la aplicación del algoritmo.

Muestra	Prediccion
Muestra 14	Nemrut Dag 1
Muestra 12	Nemrut Dag 0
Muestra 13	Gollu Dag 0
Muestra 10	Bingol B 0
Muestra 11	Nemrut Dag 0
Muestra 4	Nemrut Dag 0
Muestra 5	Nemrut Dag 0
Muestra 6	Bingol B 0
Muestra 7	Bingol B 0
Muestra 1	Nemrut Dag 0
Muestra 2	Nemrut Dag 0
Muestra 3	Bingol B 0
Muestra 8	Nemrut Dag 0
Muestra 9	Nemrut Dag 0

AP	CA	Sensitivity	Specificity
0.879	0.885	1.000	1.000

Fig. 4.23. Resultados predicción y validación árbol de decisión

4.7.5. Implementación *Random Forest*

Para la adición al programa de esta técnica se procedió a la creación de un diálogo en el que se podía modificar varios parámetros. Por una parte se encuentran:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Selección de muestras.* Permite seleccionar las muestras con las que se quiere trabajar.
- *Número de árboles.* Permite indicar el número de árboles con el que trabaja el algoritmo.
- *Splitter*. Permite seleccionar el tipo de *splitter* deseado.
- *Descender.* Permite seleccionar el tipo de *descender* deseado.
- *Profundidad.* Permite limitar la profundidad máxima del árbol resultante.
- *Diferenciar entre subgrupos dentro de una misma fuente.*

- *Cross validation.* Técnica de validación del modelo.

Por otra parte admite indicar el procesamiento de datos deseado:

- Eliminar compuestos que no contiene la muestra.[2]
- Eliminar compuestos que no explican la fuente.

En la Figura 4.24 se observa el resultado final del diálogo.

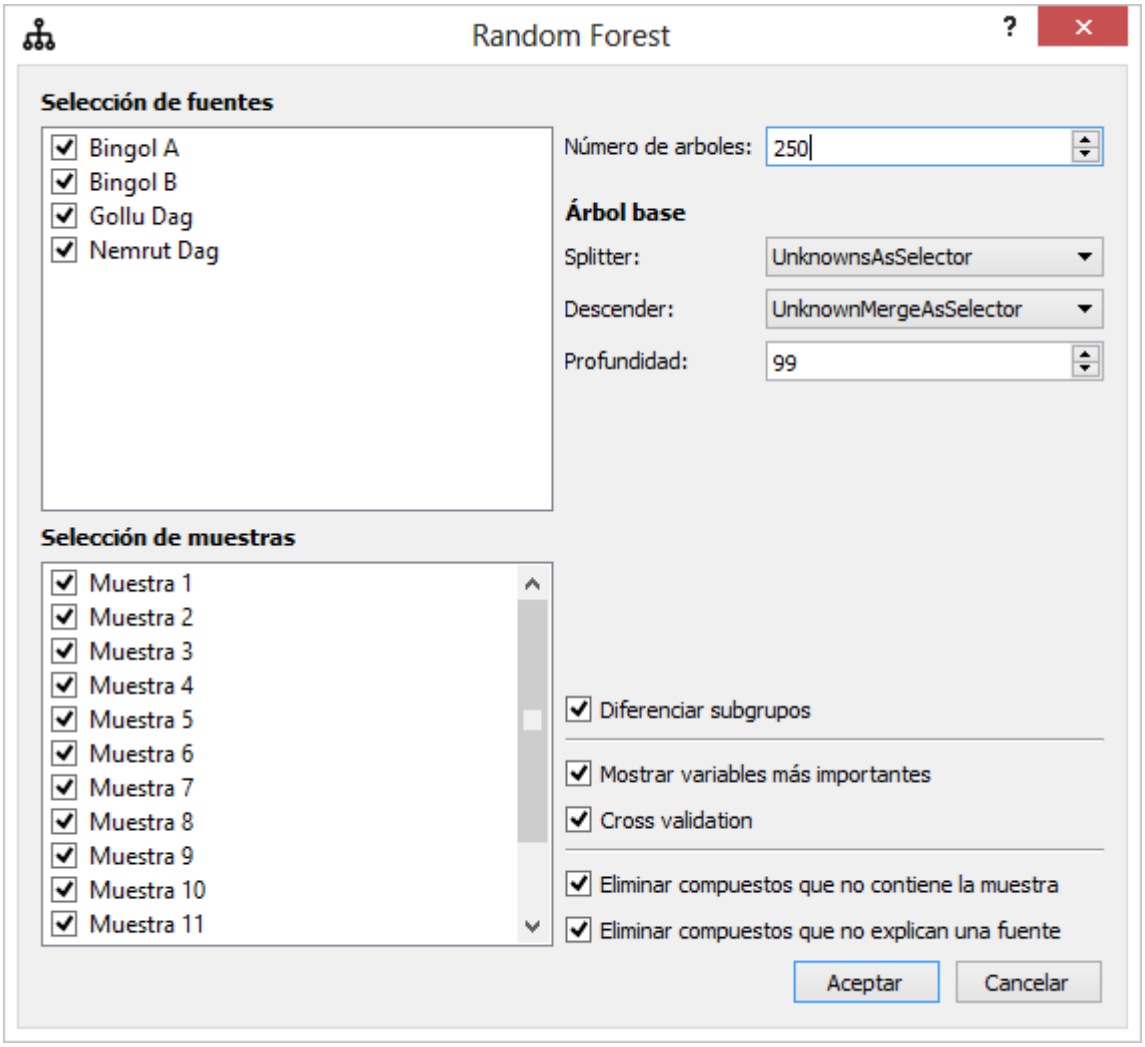


Fig. 4.24. Diálogo *Random Forest*

Como ya se ha comentado, *Random Forest* otorga la posibilidad de calcular las variables que adquieren más importancia en el proceso de predicción. En la Figura

[2] No se recomienda realizar análisis incluyendo muestras que tengan compuestos muy dispares entre ellas. Esto reduce significativamente el número total de variables con las que trabaja el algoritmo.

4.25 se observa un ejemplo de su aplicación. Cabe remarcar que el cálculo de estos valores requiere un tiempo relativamente elevado, siendo cada vez mayor a medida que entran más variables en juego. Tener en cuenta que realizar este cálculo y trabajar con un gran número de árboles conlleva tiempos de ejecución muy altos.

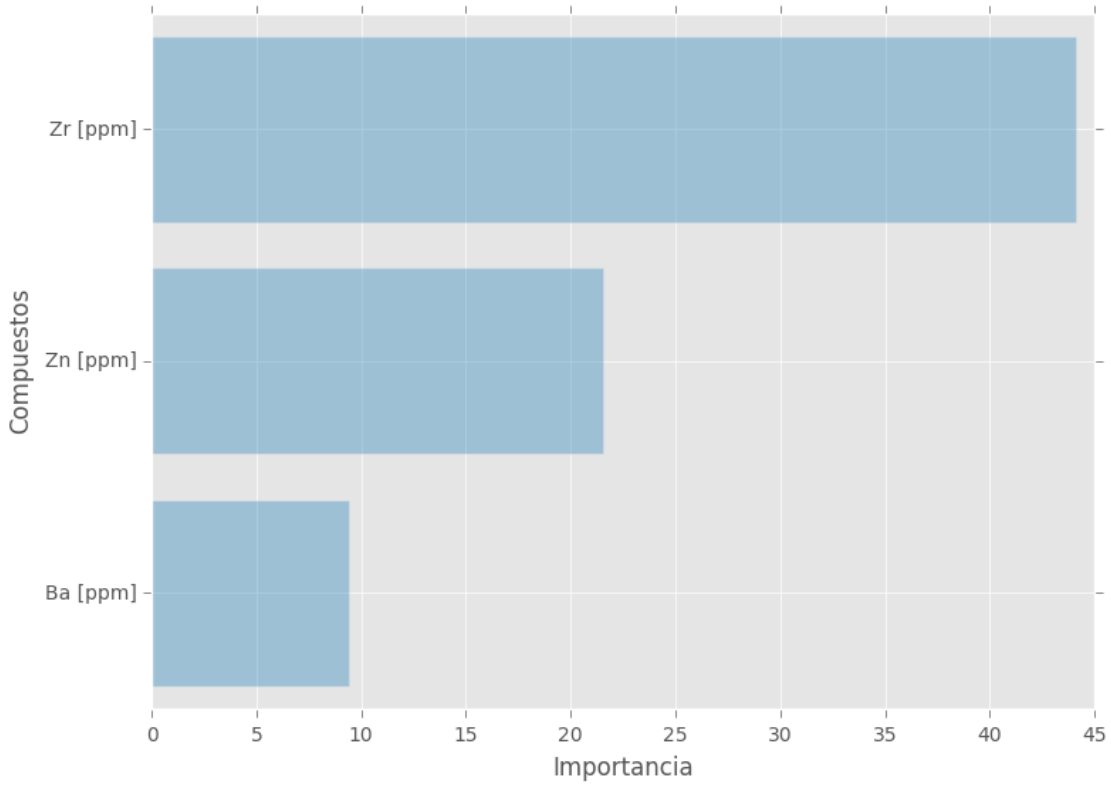


Fig. 4.25. Variables más importantes

De forma totalmente análoga a los árboles de decisión, en la hoja de sesión se muestran los resultados de predicción y validez (véase Figura 4.26).

Muestra	Prediccion
Muestra 14	Bingol A 0
Muestra 12	Nemrut Dag 0
Muestra 13	Gollu Dag 0
Muestra 10	Nemrut Dag 0
Muestra 11	Nemrut Dag 0
Muestra 4	Nemrut Dag 0
Muestra 5	Nemrut Dag 0
Muestra 6	Nemrut Dag 0
Muestra 7	Nemrut Dag 0
Muestra 1	Nemrut Dag 0
Muestra 2	Nemrut Dag 0
Muestra 3	Nemrut Dag 0
Muestra 8	Nemrut Dag 0
Muestra 9	Nemrut Dag 0

AP	CA	Sensitivity	Specificity
0.846	0.915	1.000	1.000

Fig. 4.26. Resultados de predicción y validación *Random Forest*

4.7.6. *Cross validation*

Una vez implementados los diferentes algoritmos de clasificación, fue necesario introducir técnicas de validación de modelos. De esta forma, se conseguiría obtener una idea del grado de validez de los resultados mostrados.

Se implementó como técnica de validación la técnica conocida como *cross validation,* disponible en el paquete *Orange*. Al aplicar cualquiera de los algoritmos disponibles se calculan los cuatro valores más representativos de la técnica (véase Figura 4.27):

- *AP.* Probabilidad media de ser asignado en la fuente real.
- *CA.* Porcentaje de coincidencias entre la fuente predicha y la real.
- *Sensitivity.* Proporción de positivos reales que se identificaron correctamente como tal. Resulta útil en pruebas de clasificación binaria.

- *Specificity*. Proporción de negativos que se identificaron correctamente como tal. Resulta útil en pruebas de clasificación binaria.

AP	CA	Sensitivity	Specificity
0.846	0.915	1.000	1.000

Fig. 4.27. Resultados *cross validation*

4.8. Graficado

La representación gráfica de datos también sufrió modificaciones, incluyendo nuevas funcionalidades a las ya establecidas.

En primer lugar, se incluyó la opción de permitir **seleccionar las fuentes y muestras** a representar en cada momento. Se llevó a cabo mediante la adición de dos *widgets* (*QListWidget*) al diálogo de graficado.

Además se añadió la funcionalidad de **representación de funciones matemáticas**. De este modo, se podía representar cualquier tipo de función que involucrase a uno o más compuestos diferentes. El programa debía diferenciar entre un compuesto, propiamente dicho, y el resto de la función. Por este motivo, cada uno de los compuestos que forman parte de la función deben estar envueltos por los caracteres '<>'.

Por consiguiente a la funcionalidad anterior, se modificó el método de introducción de las funciones en cada una de las coordenadas. Para ello se añadieron tres *widgets* (*QLineEdit*), uno para cada coordenada, con la funcionalidad de autocompletado. Sin embargo, el autocompletado sólo está disponible para el primer compuesto que se introduce, siendo éste uno de los posibles puntos de mejora para desarrollos futuros.

Cabe decir que el cálculo de la función se realiza gracias a la biblioteca *math* (integrada en Python). Por lo que las fórmulas introducidas deben estar en consonancia con esta biblioteca.[3]

También se incluyó la posibilidad de generar **elipses de error** indicando el *p-valor* deseado. Esta última funcionalidad sólo se implementó en gráficos bidimensionales,

[3] Consultar la documentación oficial de la biblioteca *math* para ver cómo introducir funciones compatibles: https://docs.python.org/2/library/math.html

quedando totalmente inaccesible cuando se introduce una función en la coordenada z (gráficos tridimensionales).

La distribución de elementos es tal como se muestra en la Figura 4.28.

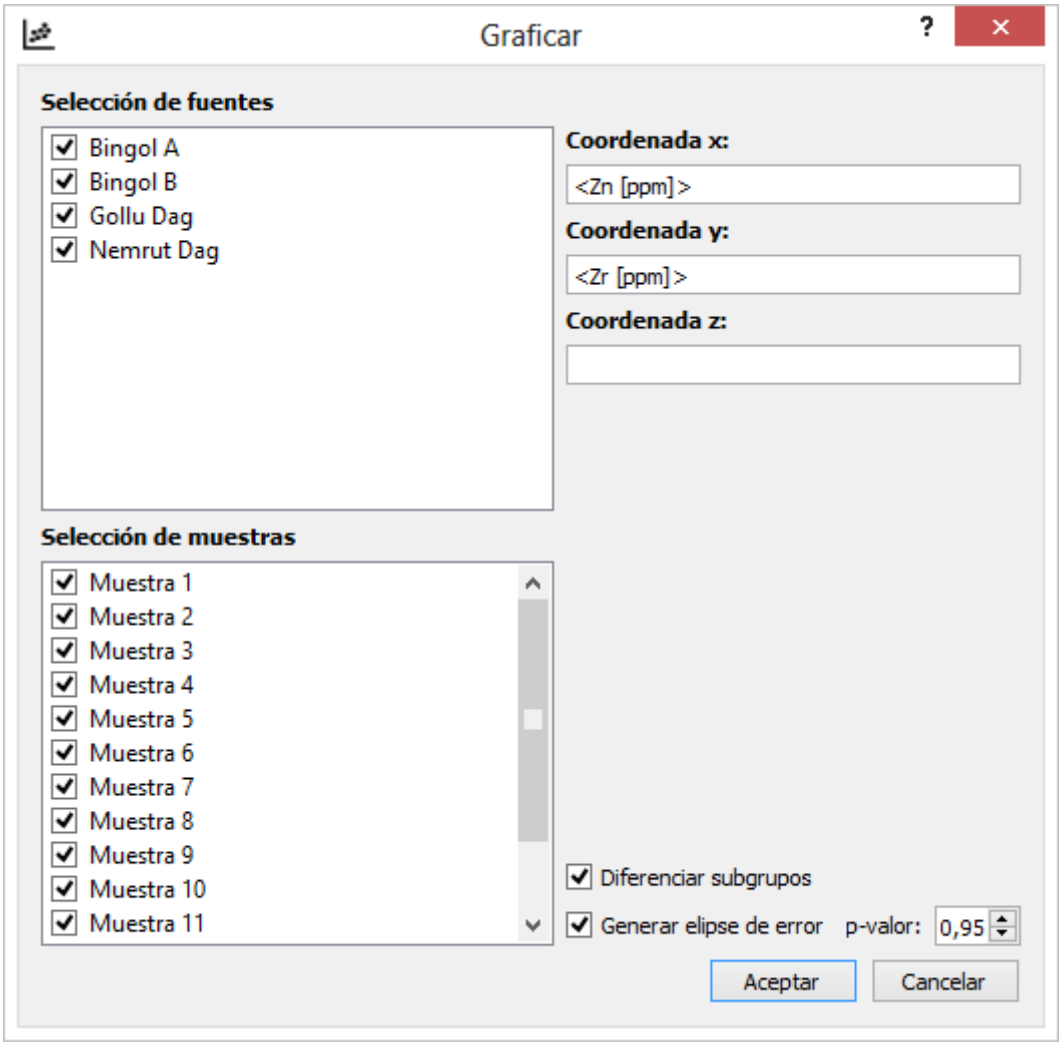


Fig. 4.28. Diálogo de representación gráfica

A continuación se observan algunos ejemplos de gráficos generados por el programa según distintas configuraciones (véase Figura 4.29 y 4.30).

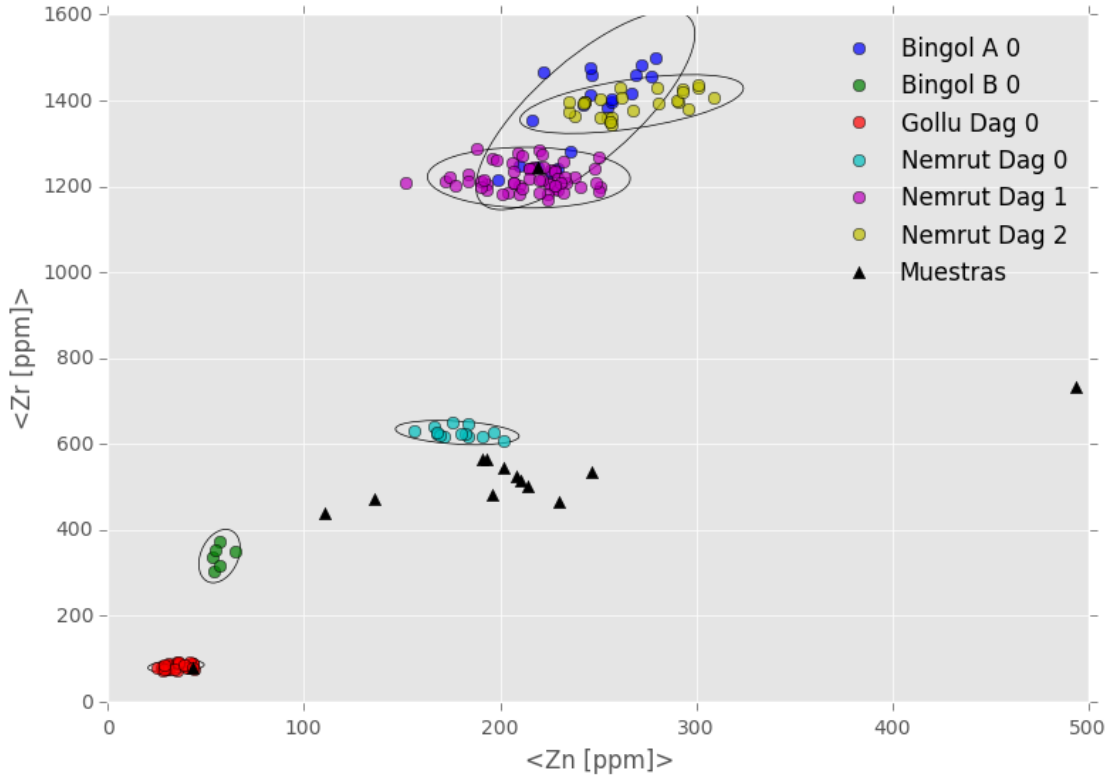


Fig. 4.29. Representación bidimensional Zn [ppm] - Zr [ppm]

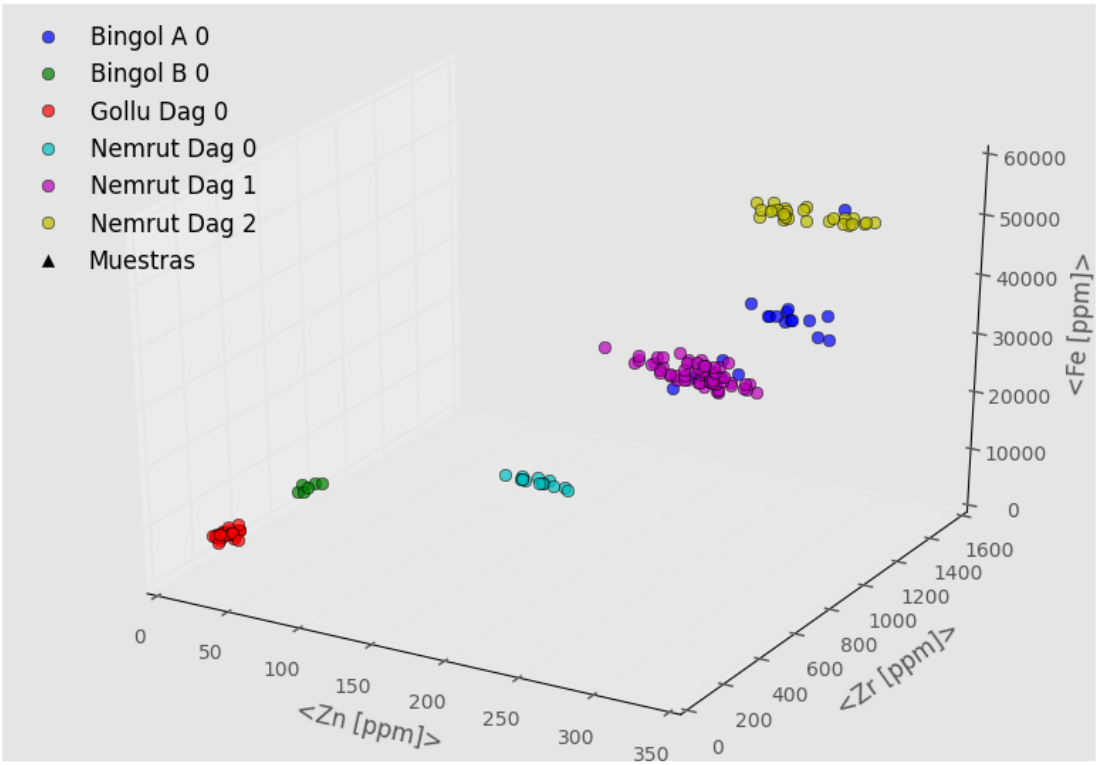


Fig. 4.30. Representación tridimensional Zn [ppm] - Zr [ppm] - Fe [ppm]

4.8.1. Cálculo elipse de error

No se encontró ninguna biblioteca externa para Python que implementará el cálculo de las elipses de error. Por ello se procedió a un cálculo manual con la ayuda de las bibliotecas *numpy* y *scipy*, siguiendo los pasos de la página web (12). De esta manera fue posible su realización e introducción posterior al gráfico.

4.8.2 Mejoría en la leyenda

Por otra parte, a petición del director del trabajo, se introdujo una pequeña mejora en la leyenda de los gráficos. El problema radicaba en que al modificar un marcador o color de un conjunto de datos en el diálogo *Figure options* del gráfico, accesible a través del mismo, la actualización de la leyenda conllevaba un marcado duplicado (véase Figura 4.31)

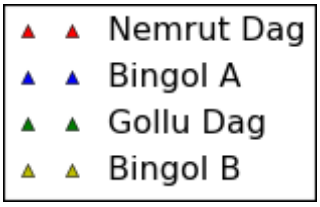


Fig. 4.31. Error en la actualización de la leyenda

La mejora se llevó a cabo a través del cambio de los parámetros por defecto (accesibles mediante el atributo *rcParams*) de la bilioteca *matplolib*. Con esto se consiguió evitar que cada vez que se actualizase la leyenda apareciese un marcador duplicado.

En la figura 4.32 se observa el diálogo *Figure options*, en el que se debe tener seleccionada la opción *(Re-)Generate automatic legend* para una correcta actualización de la leyenda.

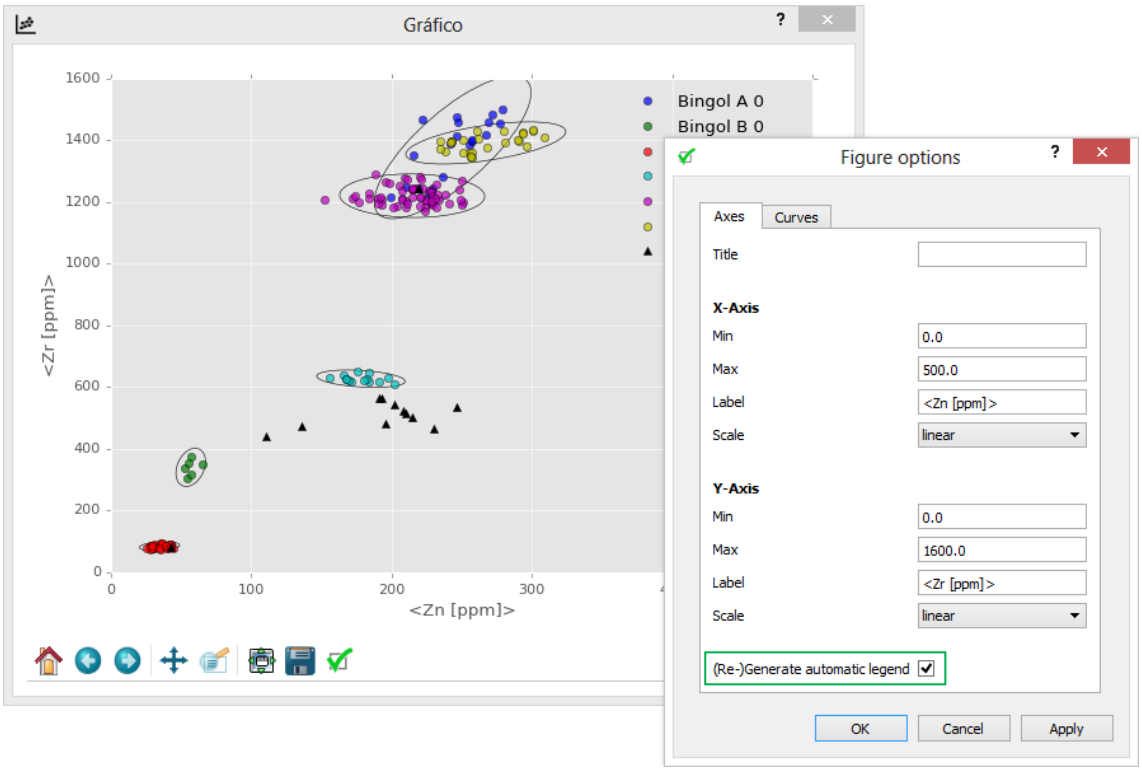


Fig. 4.32. Mejora en la leyenda

5. Vías de continuación

Dado que el proyecto está orientado a seguir siendo desarrollado en un futuro, a continuación se muestran algunas de las opciones y herramientas que, durante la realización de éste, fueron estudiadas como opciones alternativas o complementarias y que podrían resultar orientativas para futuros desarrollos.

5.1. Soporte para otras extensiones

Actualmente, el programa sólo tiene soporte para ficheros con extensión xls (Libro de Excel 97-2003), por lo que la nueva extensión xlsx de Excel no es compatible. Esta limitación viene dada por lo módulos de escritura (xlwt) y lectura (xlrd) que son utilizados para el desarrollo del programa.

Aunque esto no supone un problema, ya que siempre puede pasarse de un tipo de extensión a otra, sí resultaría interesante compatibilizar la importación de ficheros con esta nueva extensión. Para ello, se debería migrar a otros paquetes de Python de lectura y escritura de Excel que fuesen compatibles con todas las extensiones existentes.

Otra opción interesante sería compatibilizar también la importación de ficheros con extensión csv (*comma-separated values*). La mayoría de herramientas de ofimática ofrecen la posibilidad de guardar una hoja de cálculo con este tipo de extensión. De esta manera, podríamos generar nuestros datos a través de cualquier paquete de ofimática, sin la necesidad de utilizar Excel.

El tratamiento de ficheros csv es muy simple, puede ser tratado como un fichero de texto plano donde cada una de las variables está separada con ‘;’.

5.2. Autocompletado y validación en el graficado

Como ya se ha comentado anteriormente, la funcionalidad de autocompletado sólo está implementada para el primer compuesto que se escribe. Esto conlleva a que el resto de compuestos que forman parte de la función tengan que escribirse manualmente, dando lugar a posibles errores de introducción. Por ello, una de las vías de mejora sería dar una funcionalidad completa al autocompletado.

Otro concepto interesante a introducir sería la validación automática de una función. De este modo, el programa certificaría la validez de la función introducida

automáticamente, permitiendo al usuario despreocuparse por la correcta o incorrecta introducción de ésta.

5.3. Barra de progreso

Ya es conocido que el tiempo de ejecución de los algoritmos aumenta drásticamente con el número de variables (compuestos) que entran en juego. Durante el cálculo, el programa queda totalmente inaccesible, entrando en modo '*No responde*'. Este hecho puede evitarse con el uso de barras de progreso.

Las barras de progreso son elementos que permiten mostrar el estado de avance de una tarea o proceso. De este modo, aunque el tiempo de cálculo no disminuye, es posible conocer el tiempo que resta de la ejecución del algoritmo en cada momento.

La librería *PyQt* (http://pyqt.sourceforge.net/Docs/PyQt4/classes.html), concretamente mediante el elemento *QProgressBar*, permite su implementación.

5.4. Selección de variables

El conjunto de datos utilizado durante todo el trabajo posee un número relativamente elevado de variables (compuestos). Existen técnicas de selección de variables utilizadas para la diminución de la dimensionalidad de un conjunto de datos hallando aquellas que son causa de la mayor parte de variabilidad.

La reducción de variables permite representar los datos en una dimensión inferior y de manera más simple y compacta. Por consiguiente, los tiempos de ejecución de los algoritmos también disminuyen (disminuye el número de variables).

La biblioteca *Orange* y *Scikit-learn* implementan varias técnicas de selección de variables que pueden resultar útiles.

5.5. Elipsoide de error

El desarrollo actual del programa sólo contempla el cálculo de elipses de error bidimensionales. Sería interesante introducir el cálculo del elipsoide de error en el espacio tridimensional. De este modo, se podría observar un tercer compuesto/función que nos diese un mayor grado de información.

Para una primera aproximación al tema consultar la página web (12).

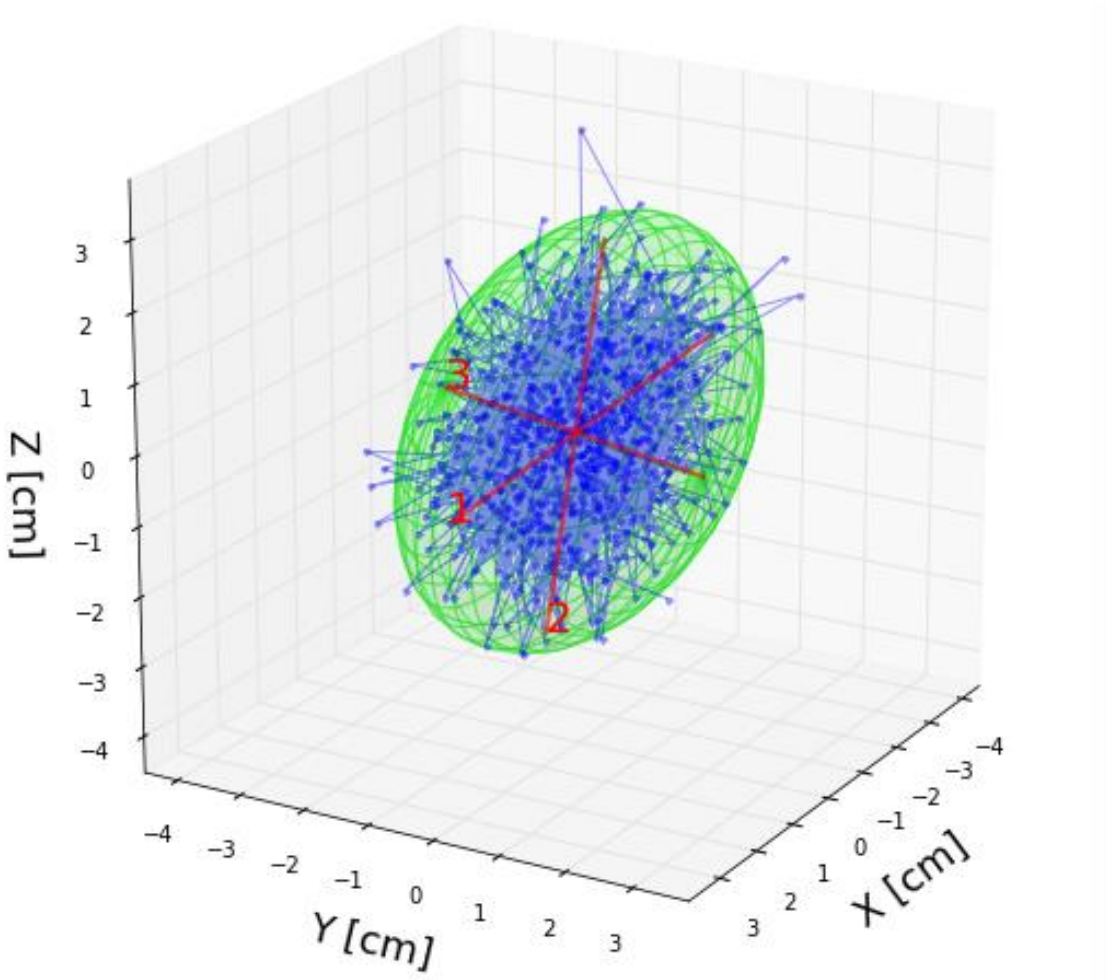


Fig. 5.1. Ejemplo de aplicación elipsoide

5.6. Mejora *Random Forest*

Como ya se ha explicado con anterioridad, el algoritmo *Random Forest* consiste en una combinación de árboles de decisión, dependiendo cada uno de ellos de una serie de variables aleatorias, donde la predicción final viene dada por la moda (valor que más se repite) de las predicciones de cada uno de los árboles individualmente.

El cálculo del número de veces que una muestra determinada se ha predicho en una fuente, puede resultar de gran ayuda a la hora de decidir sobre la certeza de una predicción. Si la mayoría de árboles realizan la misma predicción, la probabilidad de procedencia es mayor. En contraposición, si las predicciones están muy repartidas es posible realizar estudios más detallados para discernir entre las fuentes entre las que se distribuye la mayoría de predicciones (al trabajar con menos fuentes, es posible aumentar el número de variables).

5.7. *K-nearest neighbors*

El algoritmo *k-nearest neighbors* es un técnica de clasificación perteneciente a la minería de datos. Dado un conjunto de observaciones cuyas clasificaciones son conocidas, una muestra se clasificará en la clase donde se encuentre la observación cuya distancia a la muestra sea mínima.

En el presente trabajo se hicieron pruebas de implementación de *k-nearest neighbors.* Sin embargo, los resultados obtenidos no fueron los esperados. Una previa estandarización de los datos podría aumentar la validez de la técnica.

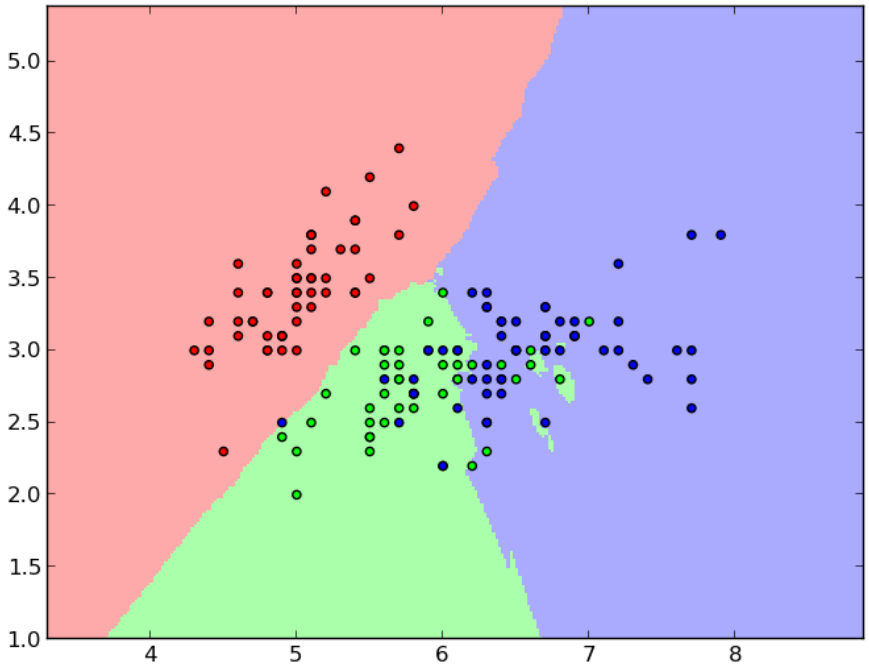


Fig. 5.2. Ejemplo de aplicación *k-nearest neighbors*

Esta técnica se encuentra totalmente implementada y parametrizable en el paquete de minería de datos *Orange*.

5.8. *Ensembles*

Los algoritmos *ensemble* utilizan múltiples modelos de clasificación para mejorar el rendimiento de predicción. El módulo *ensemble* del paquete *Orange* implementa varios enfoques al respecto, incluyendo:

- *Bagging*

Dado un conjunto de observaciones de entrenamiento, genera un predictor basado en la combinación de varios predictores formados a partir de una secuencia aleatoria de observaciones del conjunto de datos inicial.

- *Boosting*

Ídem al *bagging,* pero en la secuencia de observaciones del conjunto de datos inicial para la creación de un nuevo predictor, se presta más atención a aquellas en que los predictores anteriores han producido errores.

- *Stacking*

Implica la formación de un algoritmo de aprendizaje para combinar las predicciones de varios predictores.

5.9. Otros algoritmos de clasificación

El paquete *Orange* implementa varios algoritmos de clasificación que podrían resultar de útil aplicación, tanto a nivel alternativo como complementarios a los ya vistos:

- Naive Bayes classifier (`bayes`)
- Rule induction (`rules`)
- Support Vector Machines (`svm`)
- Logistic regression (`logreg`)
- Majority (`majority`)
- Lookup classifiers (`lookup`)
- Classifier from variable
- Constant Classifier
- Neural Network Learner (`neural`)

6. ESTUDIO Y RESULTADOS

Con el programa ya finalizado, se realizó el estudio de procedencia a partir de los datos proporcionados por el director del trabajo.

Aunque dichos estudios ya fueron realizados por el proyectista anterior, volvió a llevarse a cabo para comprobar la validez y el funcionamiento práctico de las nuevas técnicas implementadas.

6.1. Caracterización de las fuentes

En primer lugar, se decidió realizar una caracterización de las fuentes con los datos originales. El conjunto de datos inicial estaba formado por 174 muestras que provenían de 4 fuentes distintas: Göllü Dağ, Bingöl A, Bingöl B y Nemrut Dağ.

Antes de aplicar el algoritmo de agrupamiento *k-means*, se realizó un estudio previo de los datos. Se optó por buscar las mejores combinaciones según el **criterio de la *PseudoF*** en el caso simple (agrupaciones de dos elementos químicos). De este modo, se podría apreciar gráficamente los posibles subgrupos de cada una de las fuentes. Como condición, se forzó a que apareciesen todos los volcanes y a una estandarización previa de los datos con la intención de incluir el máximo número de datos posibles.

Compuesto	Compuesto	Indice
Nb [ppm]	Y [ppm]	404.900308604
Y [ppm]	Zr [ppm]	376.497957036
Nb [ppm]	Zr [ppm]	286.849919369
Zn [ppm]	Zr [ppm]	276.396200849
Ca [ppm]	Y [ppm]	211.025392015

Fig. 6.1. Resultados mejores combinaciones simples

Las primeras combinaciones, a pesar de incluir todos los volcanes y obtener una buena discriminación, mostraban un número reducido de puntos a la hora de representarlas, especialmente la fuente Nemrut Dağ (véase Figura 6.2 y Figura 6.3).

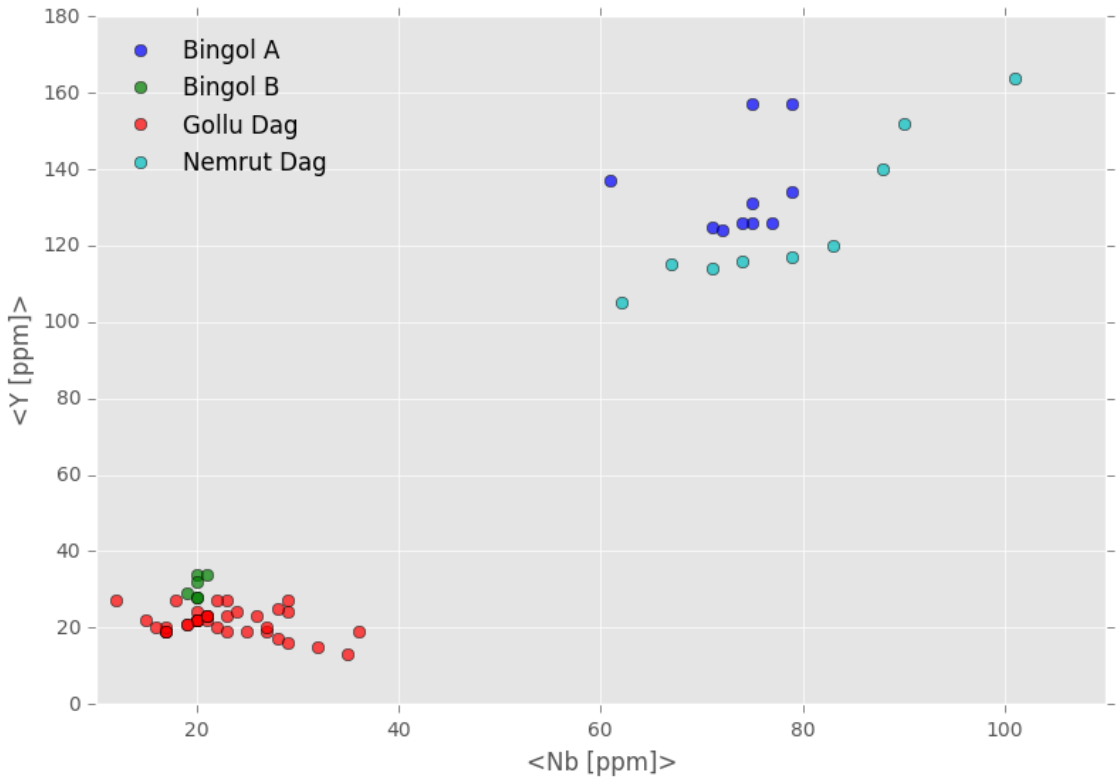


Fig. 6.2. Representación bidimensional Nb [ppm] – Y [ppm]

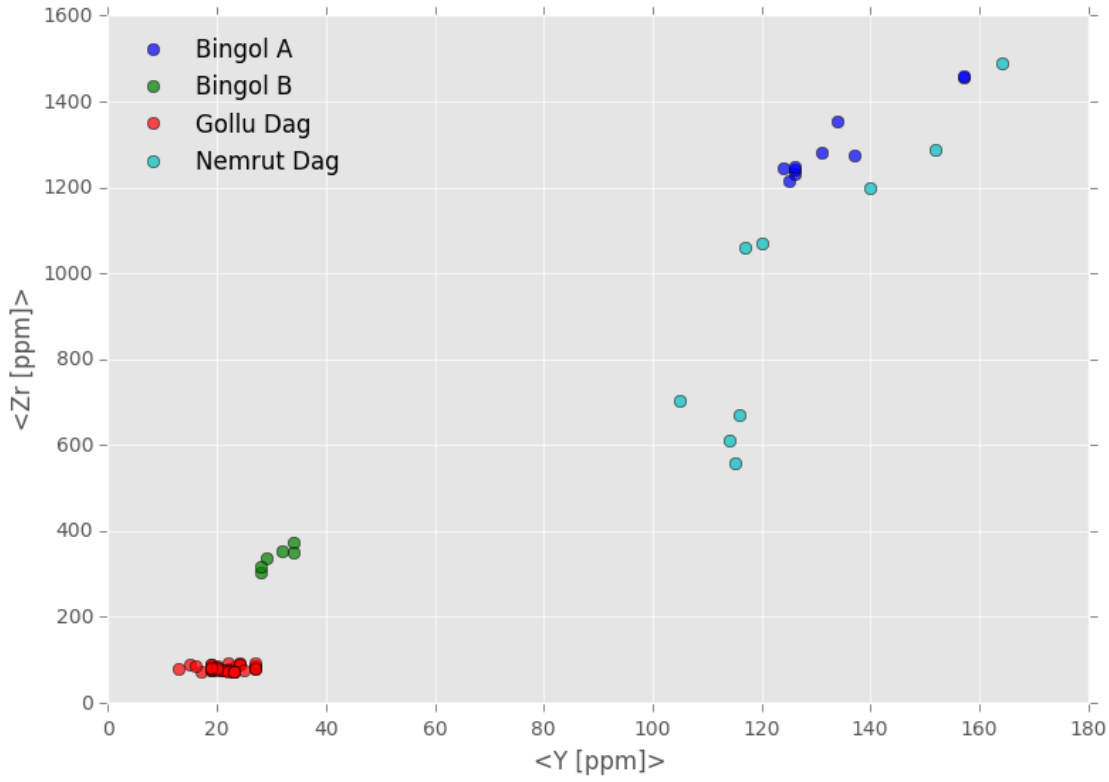


Fig. 6.3. Representación bidimensional Y[ppm] – Zr [ppm]

Un resultado particularmente útil fue el obtenido por la representación Zn [ppm] – Zr [ppm] (Figura 6.4). Se puede observar que para estos compuestos, la fuente Nemrut Dağ presenta tres subgrupos bien definidos. Para el resto de fuentes parece no haber una caracterización clara de subgrupos internos.

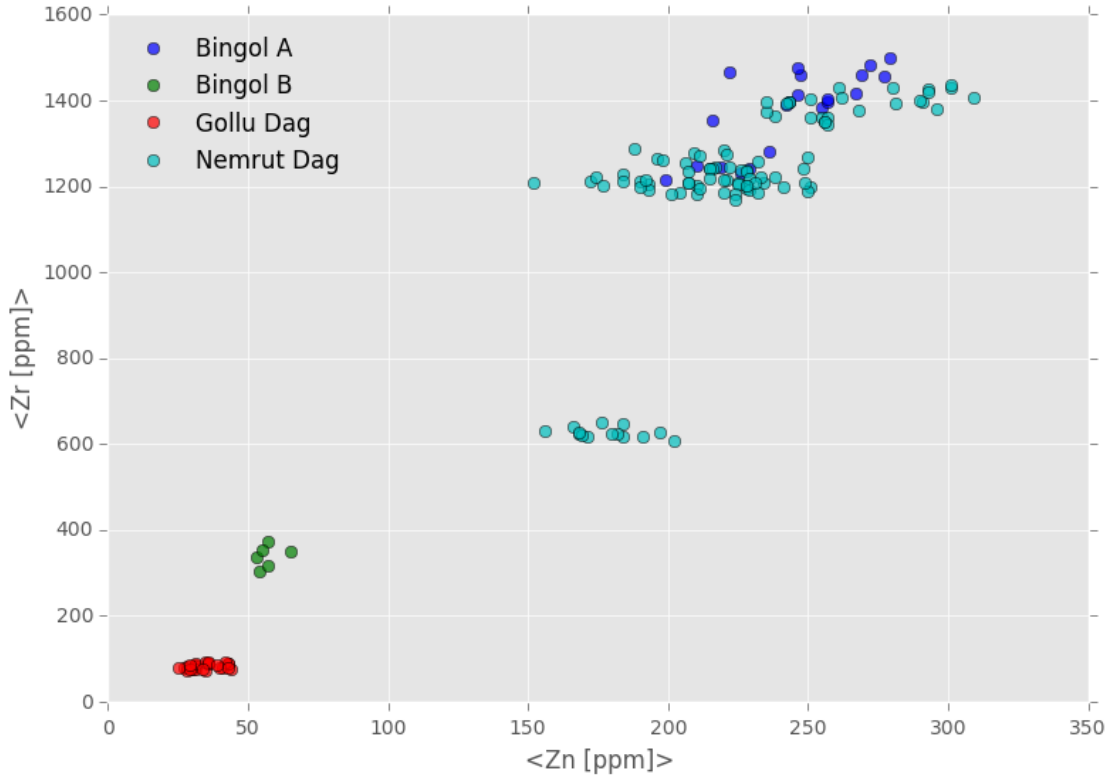


Fig. 6.4. Representación bidimensional Zn [ppm] – Zr [ppm]

Se decidió ampliar el estudio de caracterización de las fuentes con la aplicación del algoritmo ***Random Forest***. Aunque este algoritmo está más orientado a la clasificación de muestras, permite el cálculo de las variables más importantes; útil en la caracterización de las fuentes. Como condición, se forzó la eliminación de aquellos compuestos que no explicaban todas las fuentes y a una no diferenciación entre subgrupos dentro de una misma fuente. El resto de parámetros se dejaron en los valores por defecto del algoritmo. El resultado obtenido se observa en la Figura 6.5.

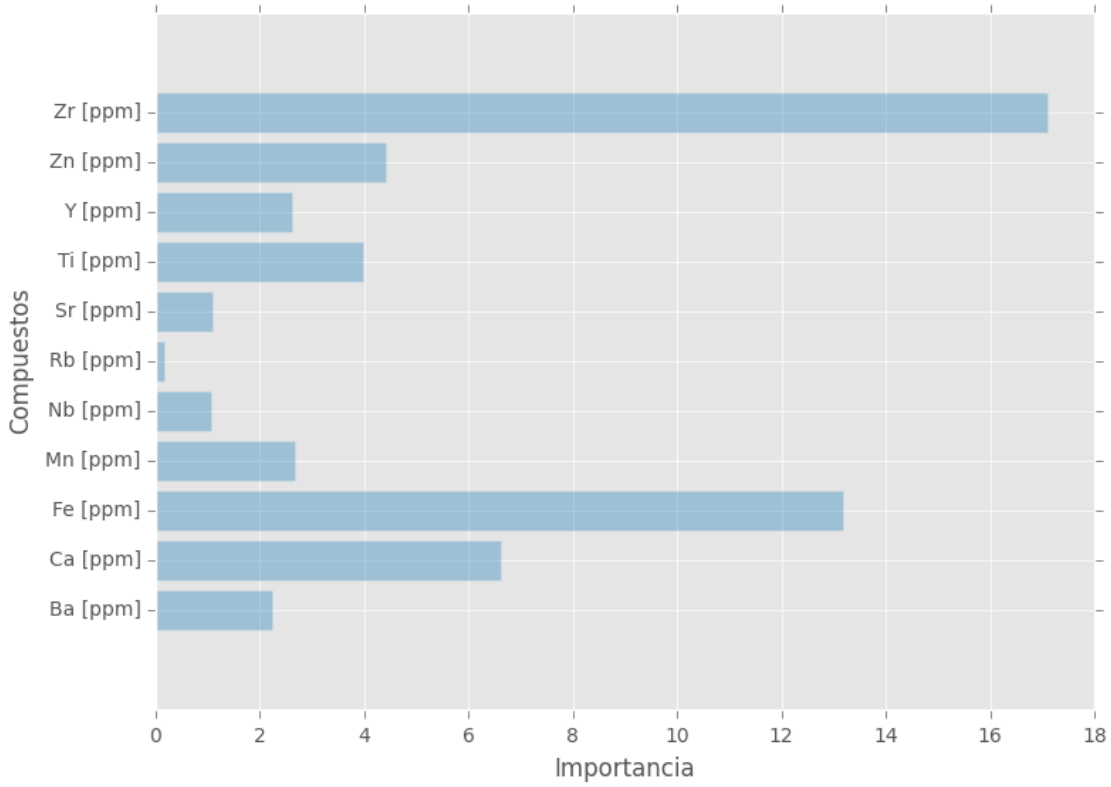


Fig. 6.5. Variables más importantes que involucran a todas las fuentes

De estos resultados se extraen las variables más características de las fuentes. Destacan las combinaciones Fe [ppm] - Zr [ppm] (Figura 6.6), Ca [ppm] - Zr [ppm] (Figura 6.7), Fe [ppm] - Ca [ppm] (Figura 6.8), Fe [ppm] - Zn [ppm] (Figura 6.9), Ca [ppm] - Zn [ppm] (Figura 6.10).

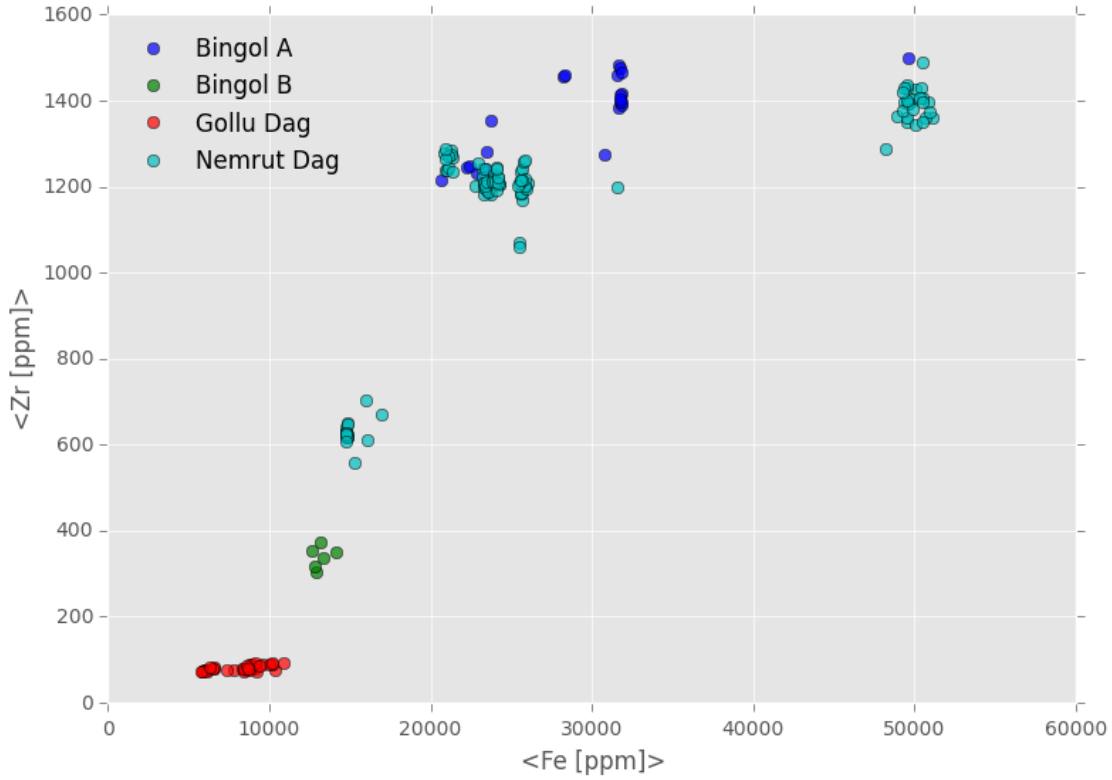


Fig. 6.6. Representación bidimensional Fe [ppm] - Zr [ppm]

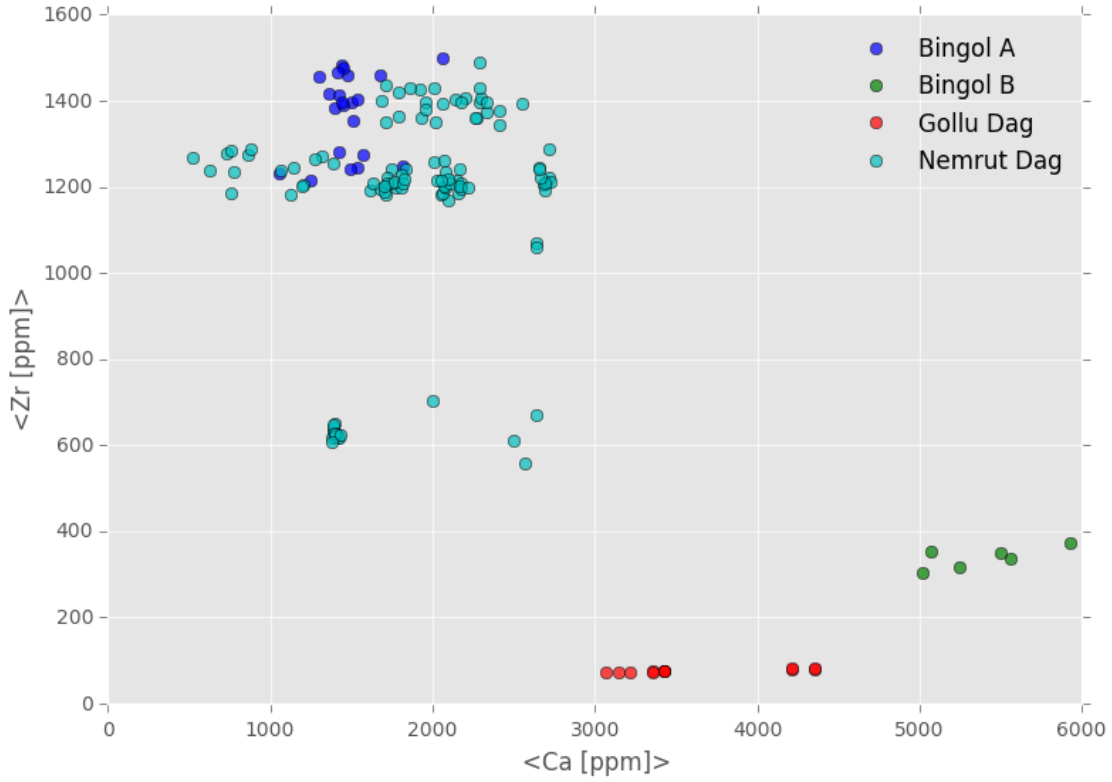


Fig. 6.7. Representación bidimensional Ca [ppm] - Zr [ppm]

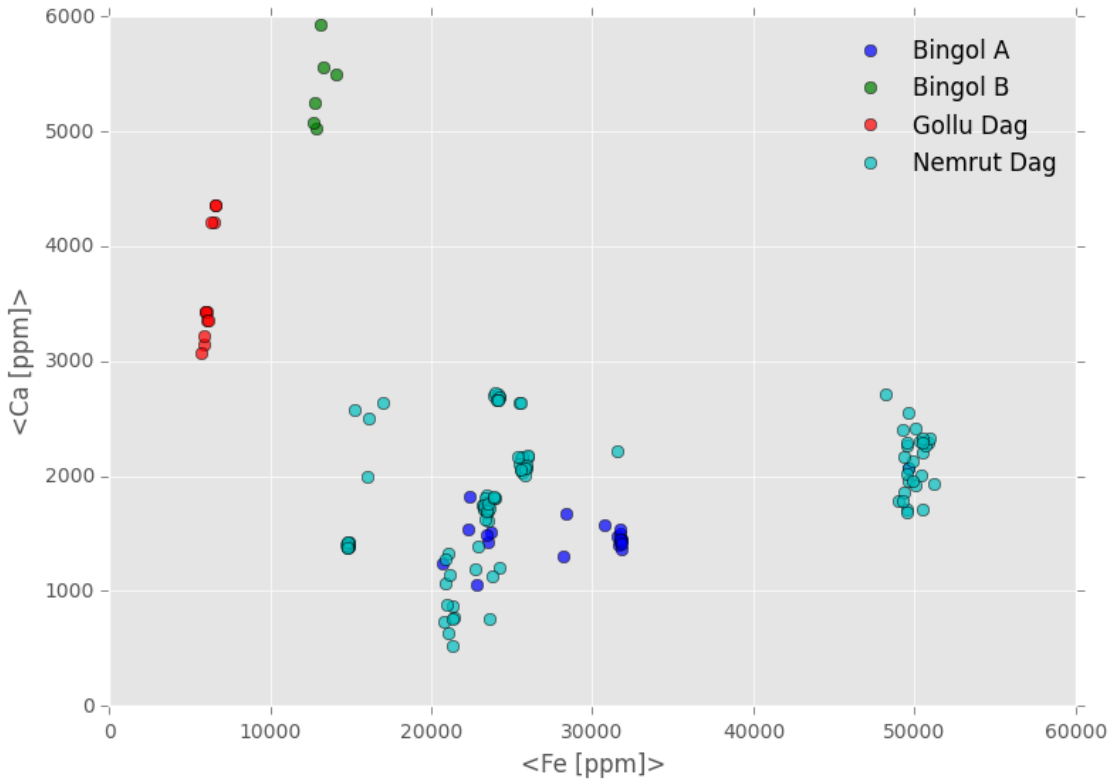


Fig. 6.8. Representación bidimensional Fe [ppm] - Ca [ppm]

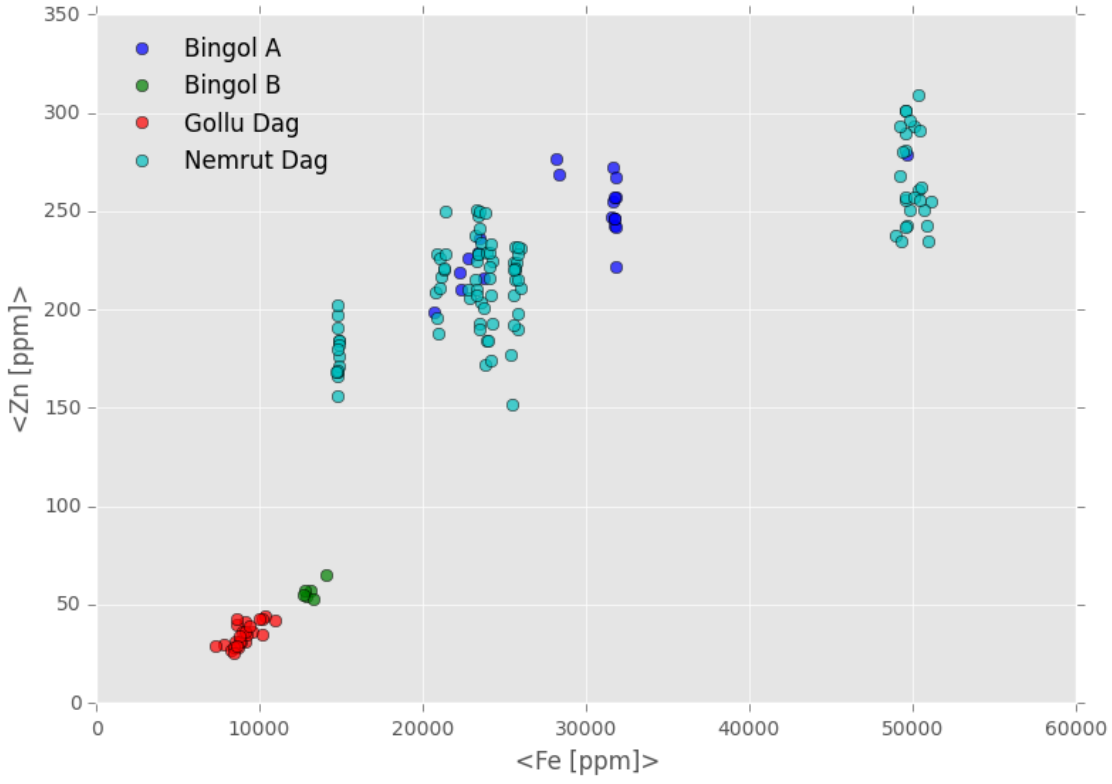


Fig. 6.9. Representación bidimensional Fe [ppm] - Zn [ppm]

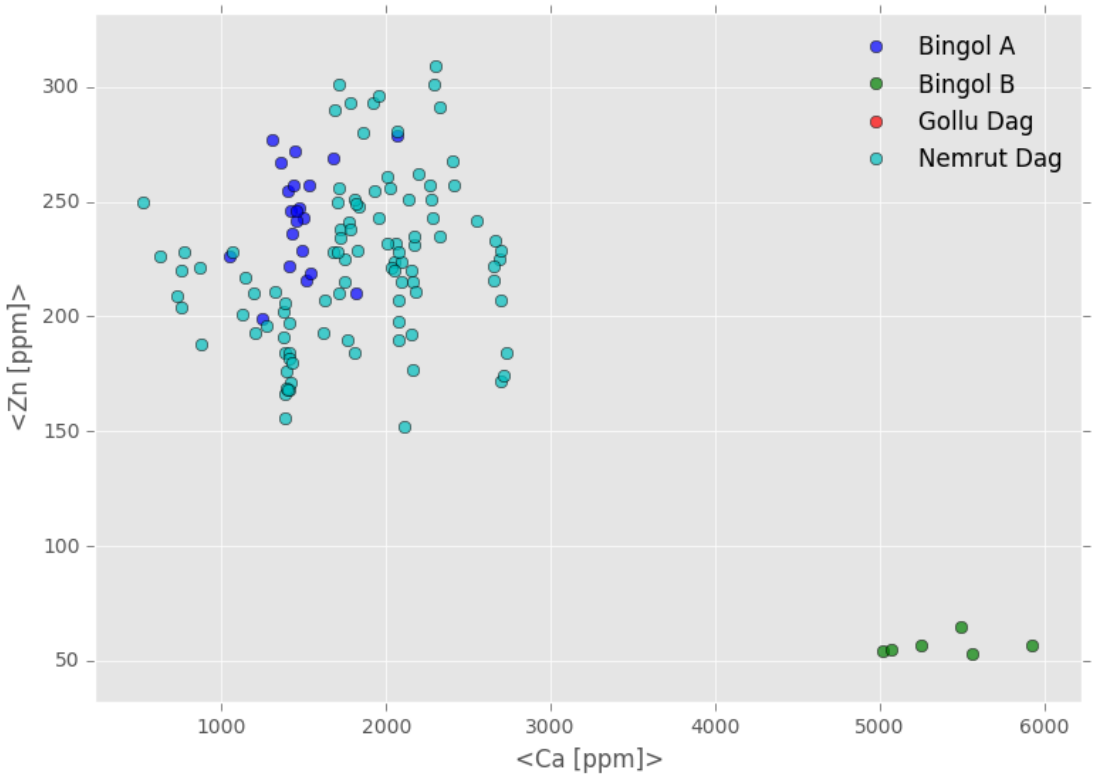


Fig. 6.10. Representación bidimensional Ca [ppm] - Zn [ppm]

De estas representaciones pueden extraerse dos conclusiones:

- Parece viable una división de tres subgrupos en la fuente Nemrut Dağ, aunque algunos compuestos no otorgan esta diferenciación tan clara.
- No parece factible la división en subgrupos en el resto de fuentes.

Una vez realizada la exploración previa de los datos, se realizó el cálculo del **coeficiente *silhouette*** para todas las fuentes con un rango de conglomerados de 2 a 8. El resto de parámetros se dejaron en los valores por defecto. Se observan los resultados de aplicación de la técnica en las Figuras 6.11 y 6.12.

Fuente	2	3	4	5	6	7	8
Bingol B	0.031	0.119	0.452	0.803	1.000	0.000	0.000
Bingol A	0.551	0.649	0.642	0.506	0.551	0.389	0.401
Nemrut Dag	0.685	0.557	0.520	0.468	0.507	0.543	0.502
Gollu Dag	0.433	0.299	0.413	0.355	0.416	0.375	0.318

Fig. 6.11. Tabla de resultados coeficiente *silhouette*

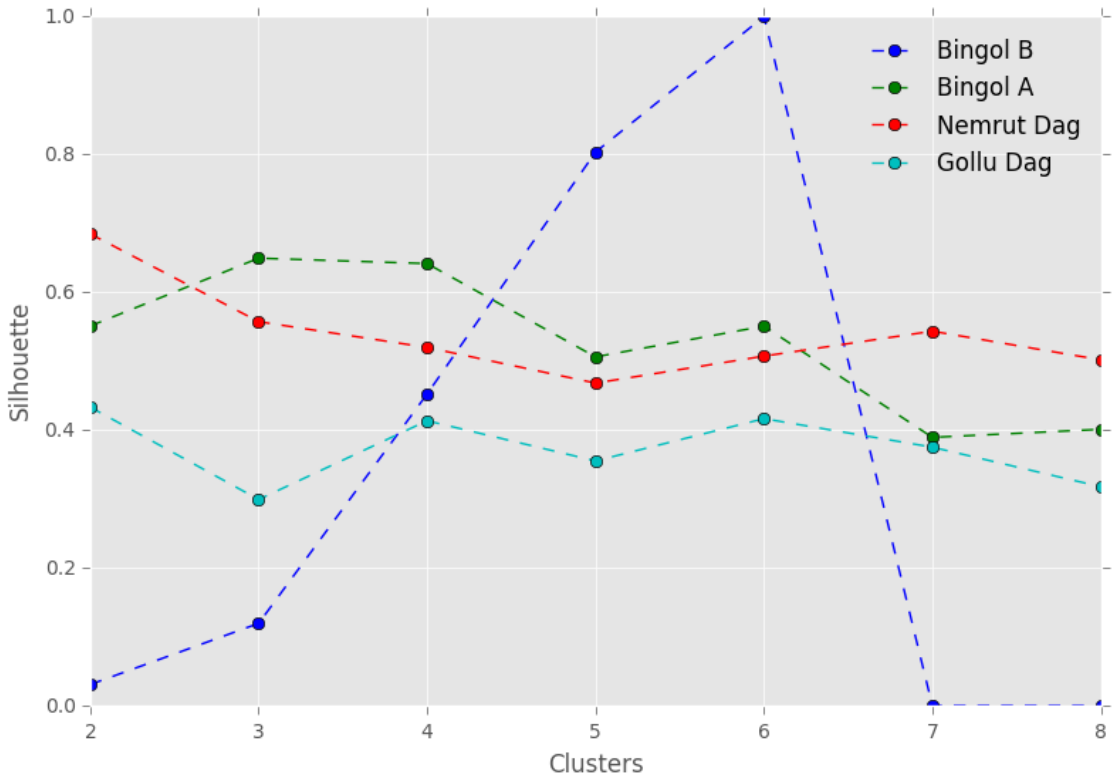


Fig. 6.12. Gráfico de resultados *silhouette*

En la Figura 6.12 se pueden ver los particulares valores que adquiere la fuente Bingöl B para los diferentes conglomerados. El conjunto de datos sólo tiene seis observaciones de la fuente Bingöl B, por lo que la opción de dividir ésta en seis subgrupos es 'ideal'.

Con el estudio previo de los datos quedó constatada la poca viabilidad de división de las fuentes Bingöl A, Bingöl B y Göllü Dağ en diversos subgrupos, por lo que el estudio se centra en los resultados obtenidos para la fuente Nemrut Dağ.

El coeficiente *silhouette* recomienda la creación de dos conglomerados para la fuente Nemrut Dağ. No obstante, se tomó la decisión de utilizar tres subgrupos tal y como se concluyó en la exploración previa de los datos. La no conciliación del coeficiente *silhouette* con el estudio visual se debe a la gran variabilidad de agrupaciones que se observan según los compuestos elegidos.

Para el cálculo de los diferentes subgrupos se procede a la aplicación del **algoritmo *k-means***. Mientras que para las fuentes Bingöl A, Bingöl B y Göllü Dağ se crea un solo conglomerado, para la fuente Nemrut Dağ se crean tres (véase Figuras 6.13 y 6.14).

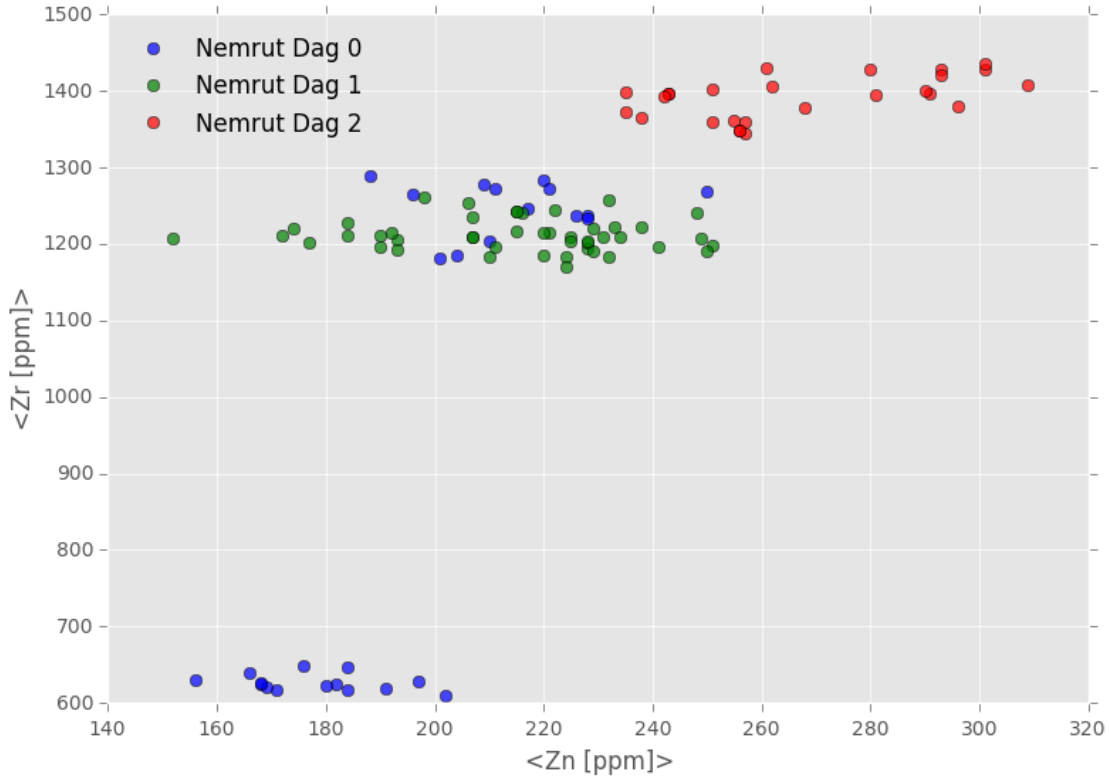


Fig. 6.13. Representación bidimensional Zn [ppm] - Zr [ppm]

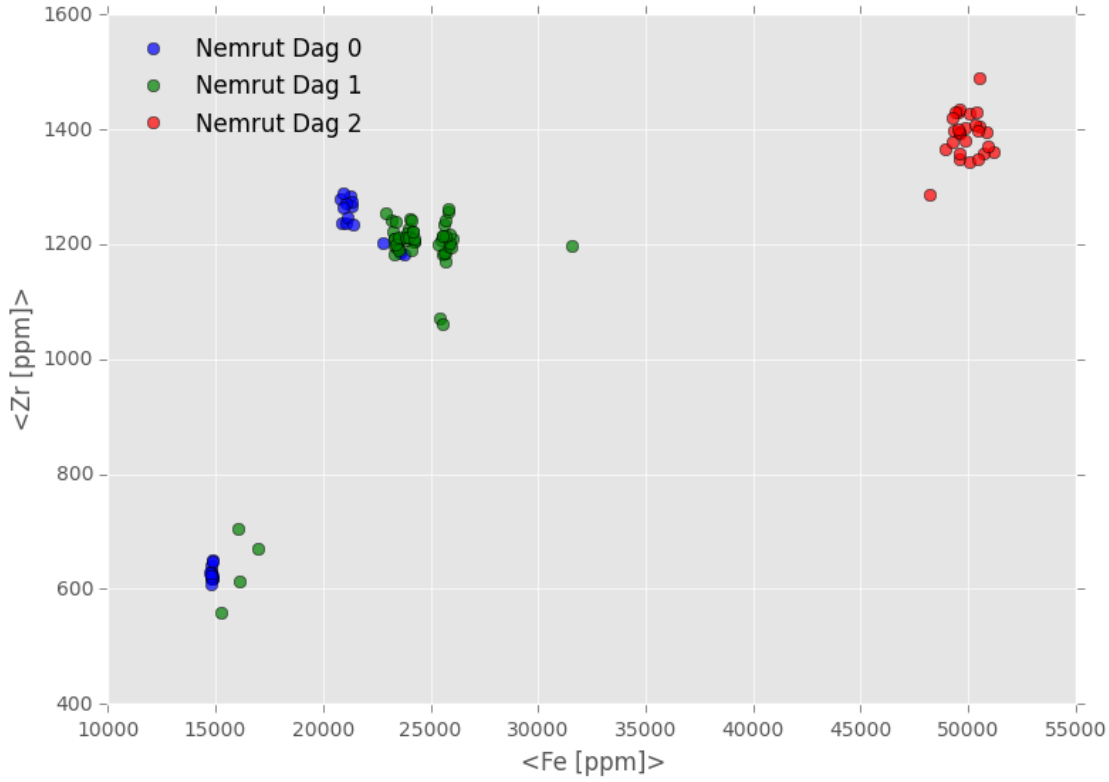


Fig. 6.14. Representación bidimensional Fe [ppm] - Zr [ppm]

En las representaciones anteriores se observa como el algoritmo *k-means* no clasifica correctamente las observaciones del conjunto de datos para la fuente Nemrut Dağ.

Recordar en este punto que, como configuración predeterminada, el algoritmo *k-means* sólo contempla aquellos compuestos que contienen el valor para todas las observaciones. Analizando detalladamente los datos, se observó que una serie de muestras de la fuente Nemrut Dağ era muy heterogénea respecto al resto de muestras; tenían muy pocos compuestos en común. Esto hacía que el algoritmo trabajase con menos compuestos y disminuyese su precisión. Por este motivo y con el fin de obtener resultados más ajustados, se buscaron configuraciones distintas a la predeterminada[4]. Por ello se decidió desmarcar la opción de eliminar todos aquellos compuestos que no explicaran todas las observaciones y cambiar el criterio de asignación por la distancia *Relief* (que incorpora el tratamiento de valores perdidos). También se modificó el método de inicialización de centroides a *init_diversity*. En la Figura 6.15 y 6.16 se observan las nuevas agrupaciones.

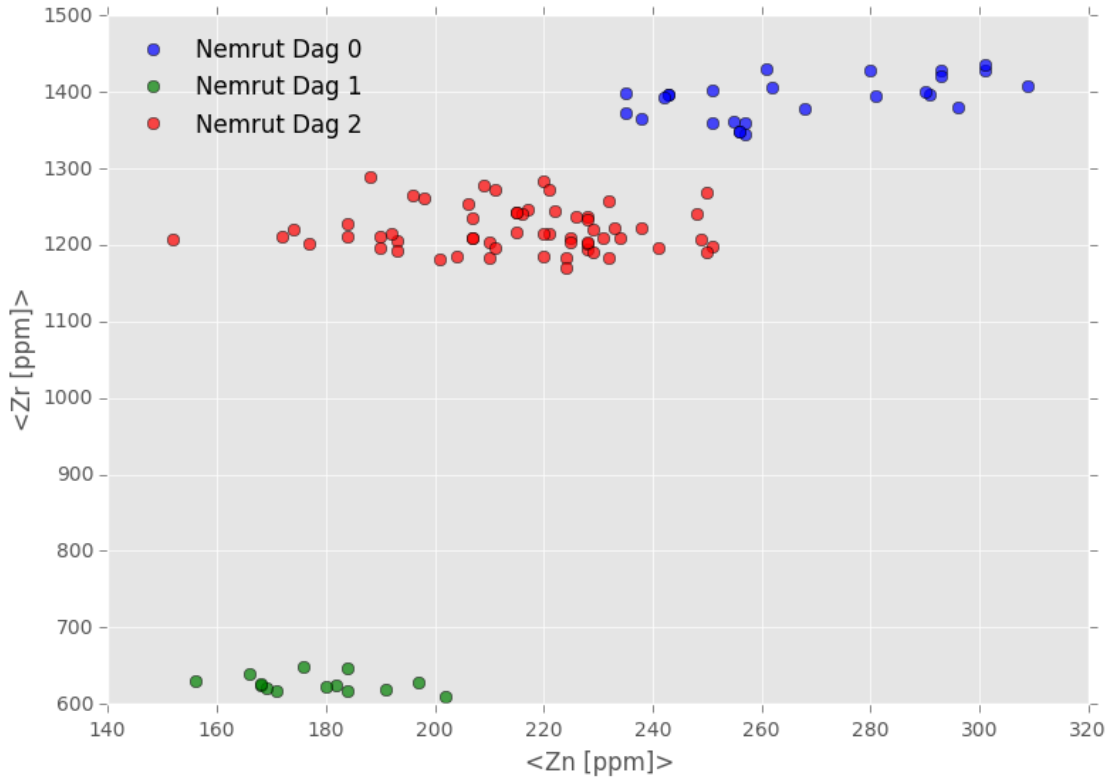


Fig. 6.15. Representación bidimensional Zn [ppm] - Zr [ppm]

[4] La eliminación de las muestras mencionadas también lleva a una correcta aplicación del algoritmo con los parámetros por defecto.

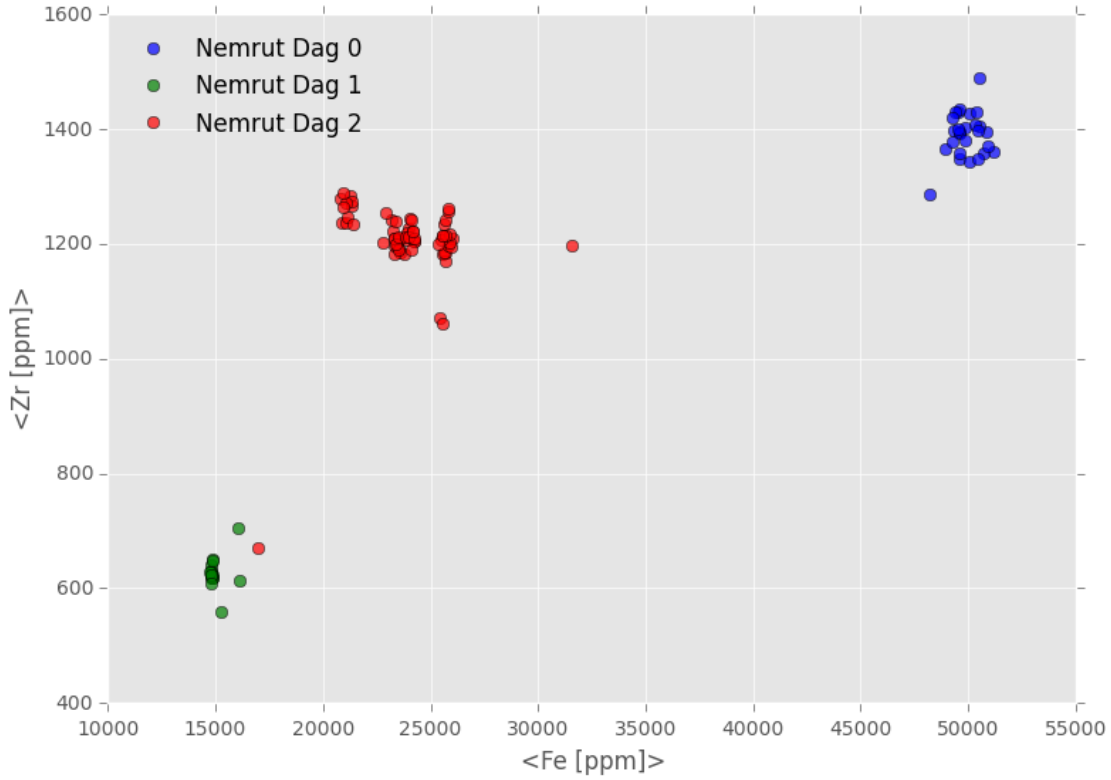


Fig. 6.16. Representación bidimensional Fe [ppm] - Zr [ppm]

Como puede observarse en la Figura 6.16, una de las muestras no está agrupada correctamente. Se procedió a cambiarle el subgrupo manualmente.

En las Figuras 6.17 y 6.18 se observan diferentes representaciones con los conglomerados formados.

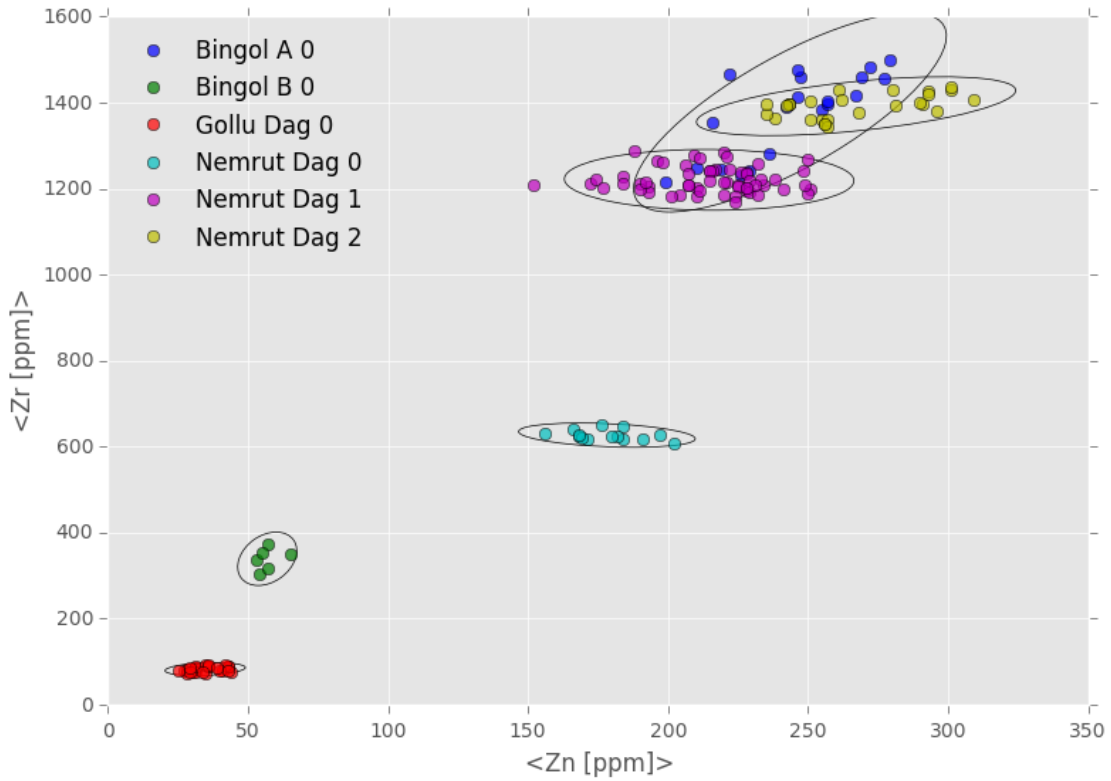


Fig. 6.17. Representación bidimensional Zn [ppm] - Zr [ppm]

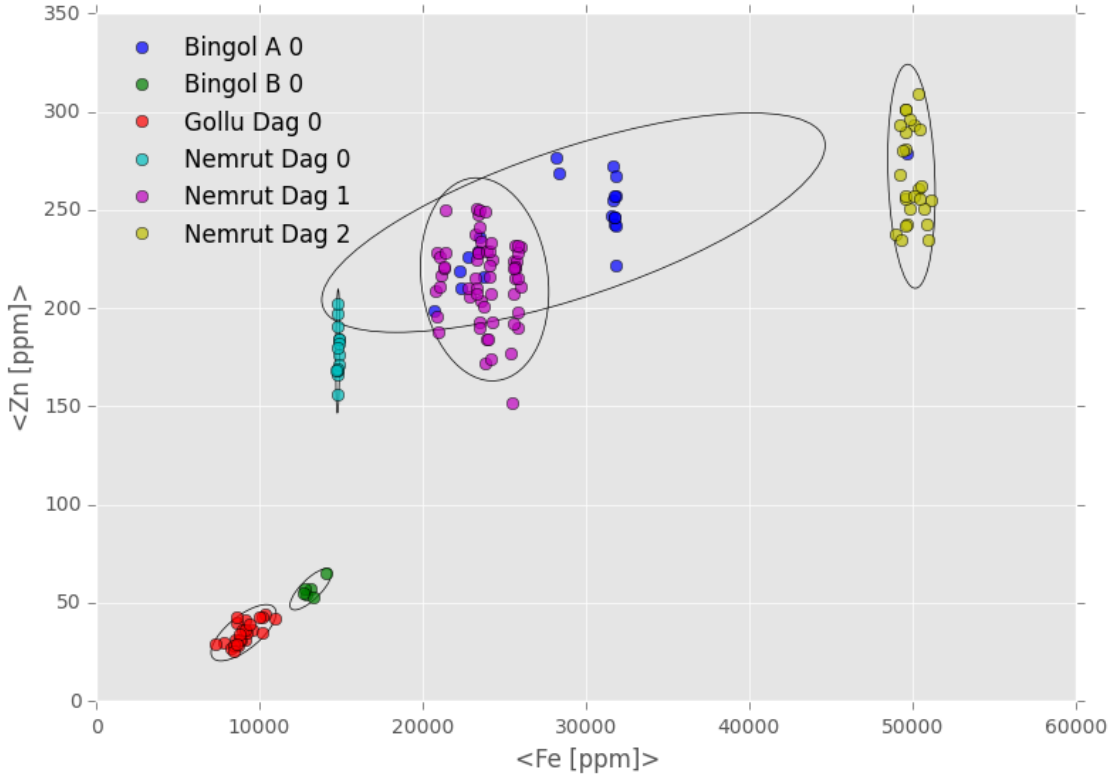


Fig. 6.18. Representación bidimensional Fe [ppm] - Zn [ppm]

6.2. Localización de yacimientos

Una vez caracterizadas las fuentes, se realiza un estudio de procedencia sobre las 12 muestras proporcionadas. Se analizaron mediante las dos técnicas de clasificación implementadas en el programa. Debido a la gran homogeneidad de las muestras (la mayor parte de estas contienen los mismos compuestos), los análisis pueden aplicarse sobre todas ellas al mismo tiempo.

Árbol de decisión

En el diálogo de configuración se seleccionaron todas las muestras y fuentes disponibles. Los parámetros *splitter* y *descender* fueron configurados como *UnknownsToBranch* y *UnknownToBranch* respectivamente (configuración por defecto) y se limitó su profundidad a cuatro. Se diferenció entre subgrupos y se eliminaron los compuestos que no contenían las muestras y/o que no explicaban alguna fuente.

En la Figura 6.19 se observan los resultados de predicción y validación del modelo.

Muestra	Prediccion
Muestra 12	Nemrut Dag 2
Muestra 10	Nemrut Dag 2
Muestra 11	Nemrut Dag 2
Muestra 4	Nemrut Dag 2
Muestra 5	Nemrut Dag 2
Muestra 6	Nemrut Dag 2
Muestra 7	Nemrut Dag 2
Muestra 1	Nemrut Dag 2
Muestra 2	Nemrut Dag 2
Muestra 3	Nemrut Dag 2
Muestra 8	Nemrut Dag 2
Muestra 9	Nemrut Dag 2

AP	CA	Sensitivity	Specificity
0.897	0.903	1.000	1.000

Fig. 6.19. Resultados árbol de decisión

En la Figura 6.20 se presenta el árbol de decisión obtenido.

```
Nb [ppm]>95.500: Nemrut Dag 2 (100.00%)
Nb [ppm]<=95.500
|     Zn [ppm]>273.000: Bingol A 0 (100.00%)
|     Zn [ppm]<=273.000
|     |     Ba [ppm]<=154.200: Bingol A 0 (100.00%)
|     |     Ba [ppm]>154.200: Bingol B 0 (100.00%)
|     |     Ba [ppm]=unknown: Gollu Dag 0 (100.00%)
|     Zn [ppm]=unknown
|     |     Sr [ppm]=unknown: Nemrut Dag 1 (100.00%)
|     |     Sr [ppm]<=1.500
|     |     |     Zr [ppm]<=887.500: Nemrut Dag 0 (100.00%)
|     |     |     Zr [ppm]>887.500: Nemrut Dag 1 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Sr [ppm]>1.500
|     |     |     Zr [ppm]<=1280.000: Gollu Dag 0 (92.31%)
|     |     |     Zr [ppm]>1280.000: Nemrut Dag 2 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
Nb [ppm]=unknown
|     Ba [ppm]>50.000: Bingol A 0 (100.00%)
|     Ba [ppm]<=50.000
|     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Zr [ppm]<=1319.000
|     |     |     Zr [ppm]<=909.500: Nemrut Dag 0 (100.00%)
|     |     |     Zr [ppm]>909.500: Nemrut Dag 1 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Zr [ppm]>1319.000
|     |     |     Zr [ppm]<=1448.000: Nemrut Dag 2 (75.00%)
|     |     |     Zr [ppm]>1448.000: Bingol A 0 (100.00%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
|     Ba [ppm]=unknown
|     |     Zr [ppm]<=1308.000: Nemrut Dag 1 (100.00%)
|     |     Zr [ppm]=unknown<null node>: <null node>
|     |     Zr [ppm]>1308.000
|     |     |     Zr [ppm]<=1397.000: Nemrut Dag 2 (100.00%)
|     |     |     Zr [ppm]>1397.000: Nemrut Dag 2 (66.67%)
|     |     |     Zr [ppm]=unknown<null node>: <null node>
```

Fig. 6.20. Árbol de decisión

Random Forest

También se aplicó la técnica *Random Forest* como algoritmo predictor. Para ello, en el diálogo de configuración se seleccionaron todas las muestras y fuentes disponibles. Se utilizaron 150 árboles de decisión para evitar tiempos de ejecución muy elevados. Los parámetros *splitter* y *descender* fueron configurados como *UnknownsAsSelector* y *UnknownMergeAsSelector* respectivamente (configuración por defecto). Se limitó la profundidad máxima a 99. Además, se hizo una diferenciación de subgrupos y se eliminaron los compuestos que no contenían las muestras y/o que no explicaban alguna fuente. Se habilitó también el cálculo de variables más importantes.

En la Figura 6.21 se observan los resultados de predicción y validación del modelo.

Muestra	Prediccion
Muestra 12	Nemrut Dag 0
Muestra 10	Nemrut Dag 0
Muestra 11	Nemrut Dag 0
Muestra 4	Nemrut Dag 0
Muestra 5	Nemrut Dag 0
Muestra 6	Nemrut Dag 0
Muestra 7	Nemrut Dag 0
Muestra 1	Nemrut Dag 0
Muestra 2	Nemrut Dag 0
Muestra 3	Nemrut Dag 0
Muestra 8	Nemrut Dag 0
Muestra 9	Nemrut Dag 0

AP	CA	Sensitivity	Specificity
0.789	0.896	1.000	1.000

Fig. 6.21. Resultados predicción y validación *Random Forest*

En la Figura 6.22 se observa el resultado del cálculo de las variables más importantes.

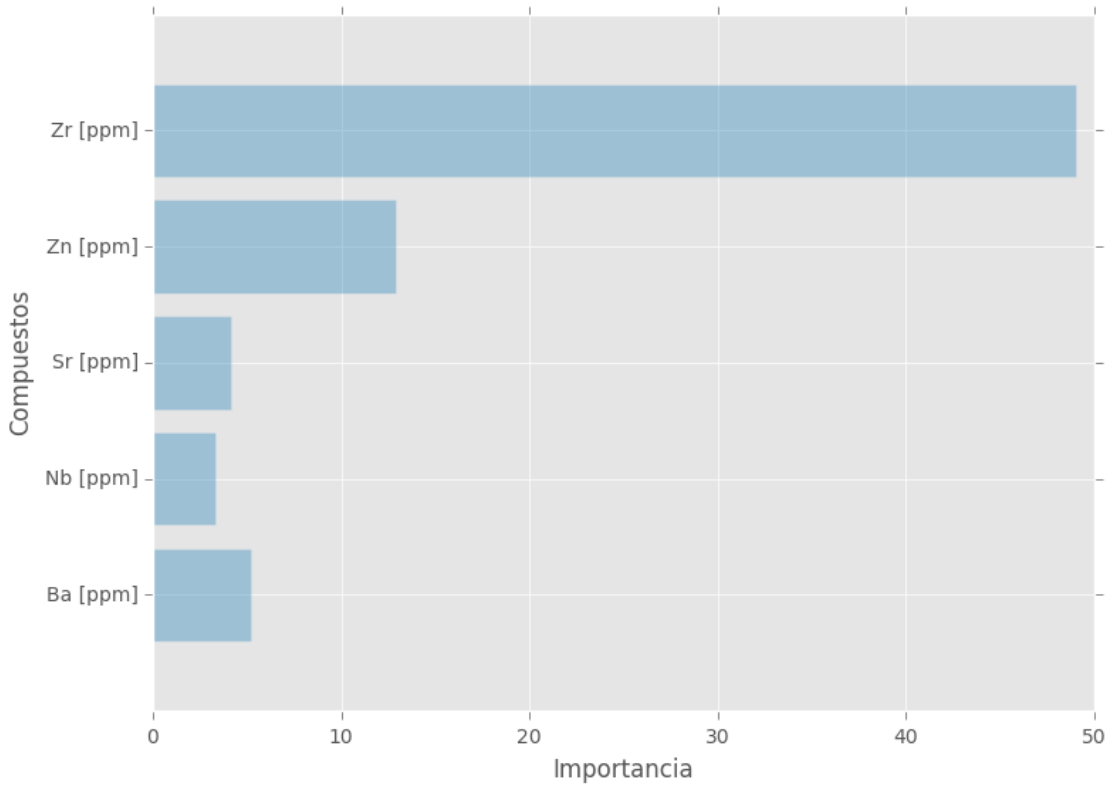


Fig. 6.22. Variables más importantes *Random Forest*

6.2.1 Conclusiones localización de yacimiento

A partir de los compuestos más característicos obtenidos en la Figura 6.22, se pueden representar toda una serie de gráficos, tanto bidimensionales como tridimensionales, que permiten valorar los resultados obtenidos. Se muestran así las representaciones Zn [ppm] - Zr [ppm] (Figura 6.23), Zr [ppm] - Ba [ppm] (Figura 6.24), Zn [ppm] - Ba [ppm] (Figura 6.25), Zr [ppm] - Nb [ppm] (Figura 6.26), Zn [ppm] - Zr [ppm] - Ba [ppm] (Figura 6.27) y Zn [ppm] - Zr [ppm] - Nb [ppm] (Figura 6.28).

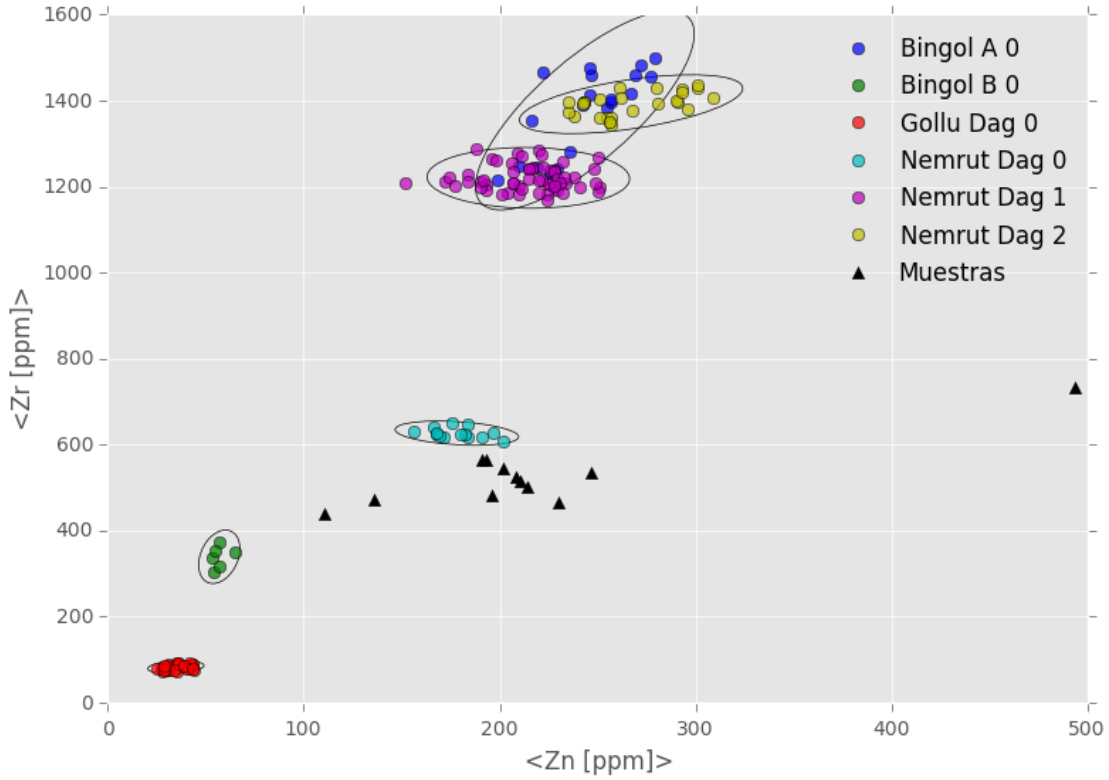


Fig. 6.23. Representación bidimensional Zn [ppm] - Zr [ppm]

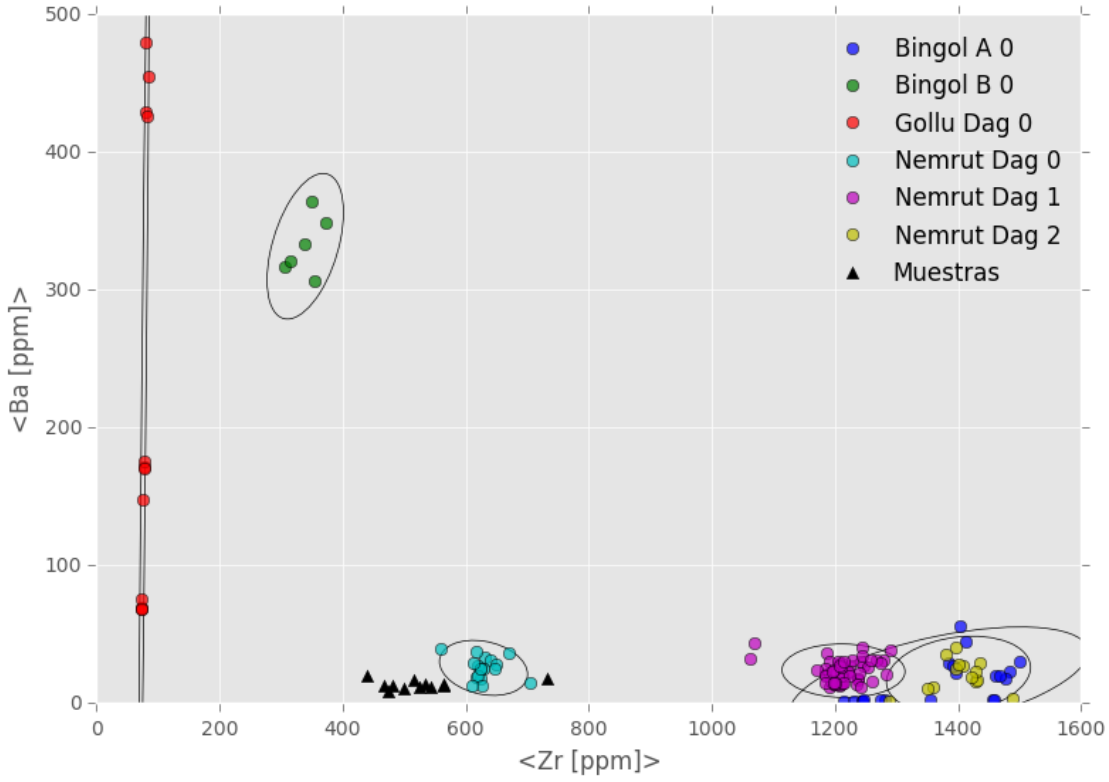


Fig. 6.24. Representación bidimensional Zr [ppm] - Ba [ppm]

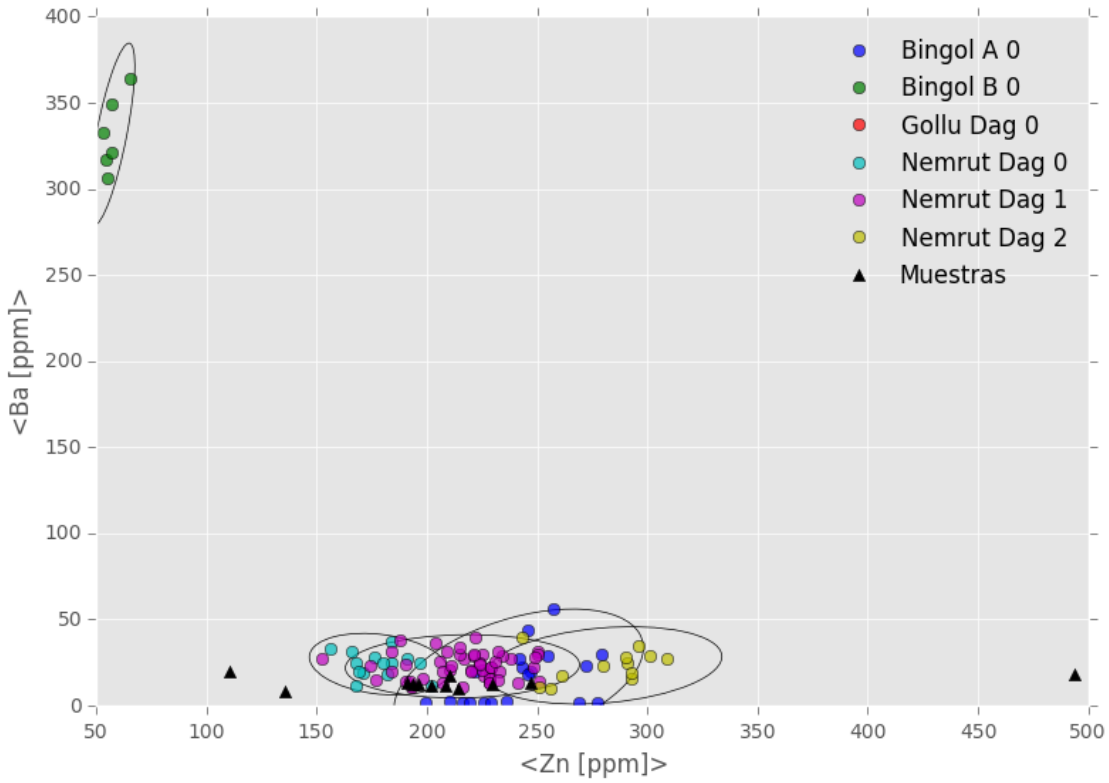


Fig. 6.25. Representación bidimensional Zn [ppm] - Ba [ppm]

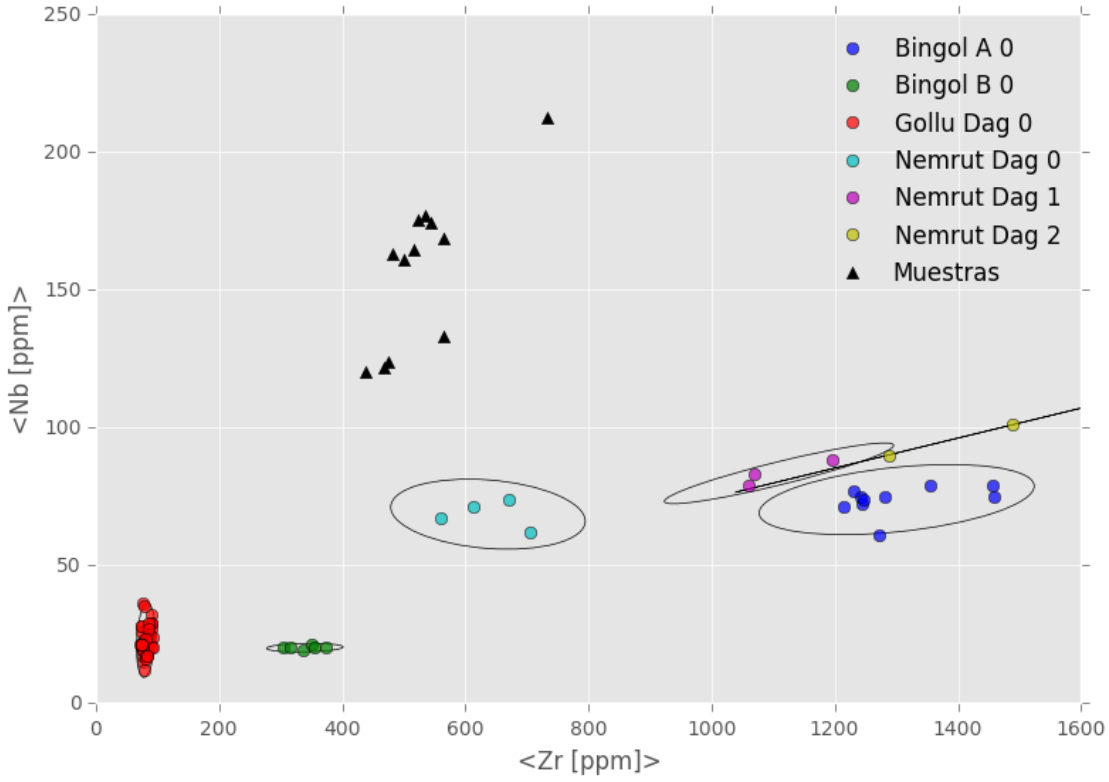


Fig. 6.26. Representación bidimensional Zr [ppm] - Nb [ppm]

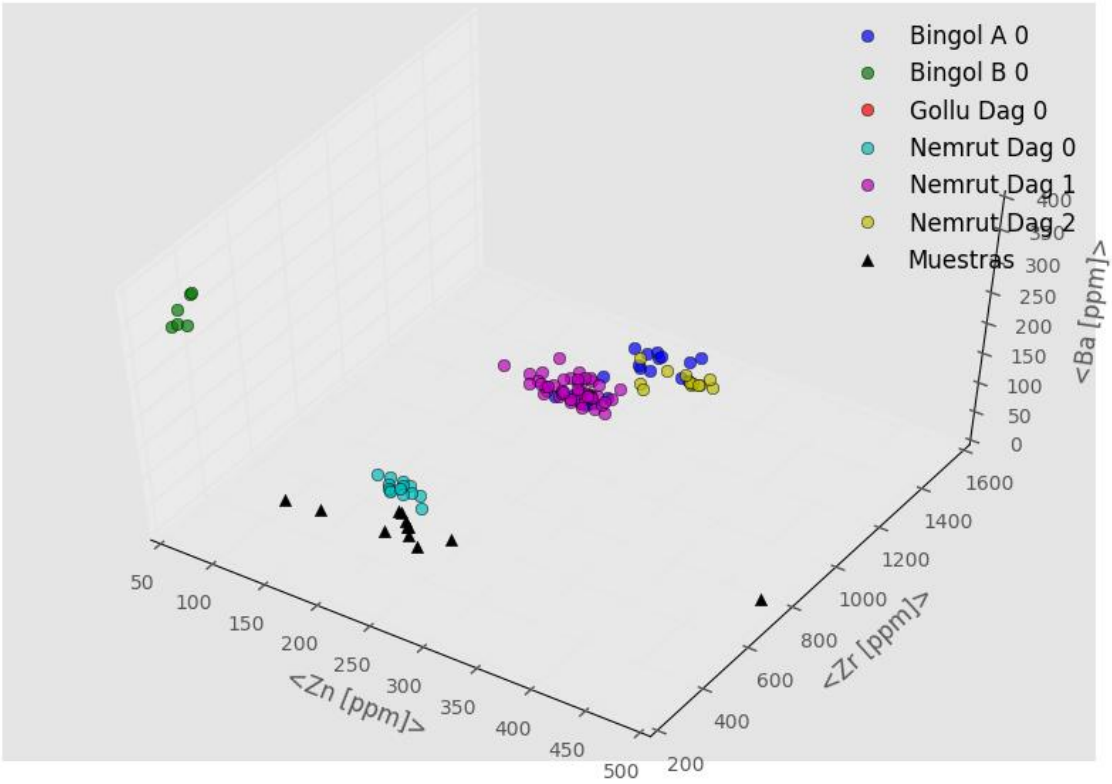


Fig. 6.27. Representación tridimensional Zn [ppm] - Zr [ppm] - Ba [ppm]

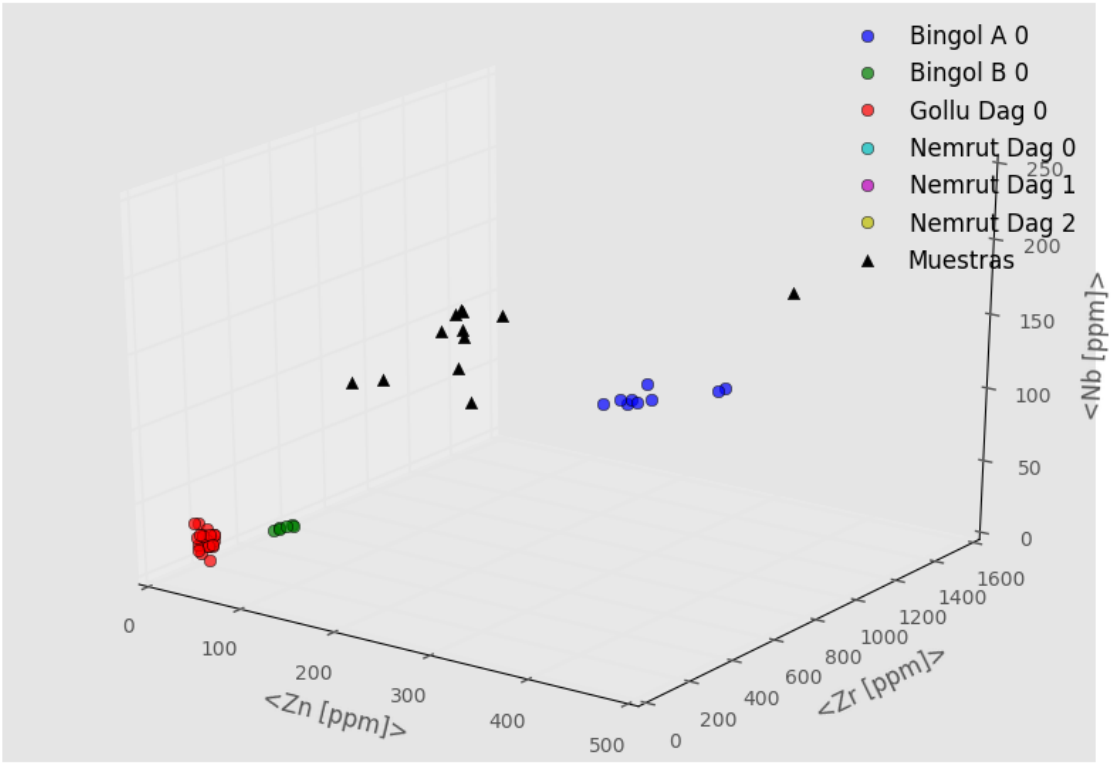


Fig. 6.28. Representación tridimensional Zn [ppm] - Zr [ppm] - Nb [ppm]

Estas representaciones, tal y como concluyó el proyectista anterior, parecen indicar que las muestras no proceden de ninguna de las fuentes estudiadas. Se puede observar como el **Niobio** (véase Figura 6.26) es el compuesto que más evidencia este hecho.

Los algoritmos de clasificación implementados llegan siempre a una solución, pudiendo ser más o menos acertada. A continuación se discute sobre las predicciones obtenidas a través de cada una de las técnicas.

Árbol de decisión

Se puede apreciar como la predicción con la configuración establecida se aleja bastante de las posibles fuentes reales de procedencia. En contraposición, se obtienen buenos parámetros de validación. Como cabía esperar, el algoritmo no está funcionando correctamente, creando un modelo sobreajustado que no generaliza bien el conjunto de datos inicial.

Si analizamos el árbol de decisión obtenido e intentamos clasificar las muestras manualmente, se observa que sólo se utiliza un único compuesto (Nb) para la predicción, poniendo de manifiesto el problema introducido en el párrafo anterior.

Se probaron configuraciones distintas modificando los parámetros *splitter* y *descender*, sin embargo, los problemas seguían presentes (la predicción se basaba en la composición de uno o dos compuestos).

Random Forest

Random Forest ofrece unos resultados mucho más ajustados a la realidad. En general, las representaciones evidencian que la fuente más cercana a la mayoría de muestras era el Nemrut Dag 0. El Niobio pone de manifiesto que las muestras no pertenecen al subgrupo establecido, sin embargo, cabe remarcar las pocas muestras que se tienen de este compuesto para la fuente Nemrut Dağ.

7. Costes

El coste económico de la realización de este proyecto se puede descomponer en tres categorías de gastos: **costes de la mano de obra, costes asociados a su realización y costes eléctricos**.

En la Tabla 7.1 se muestra el desglose de costes para la realización del trabajo.

MANO DE OBRA				
Concepto	**Coste hora**	**Horas**		**Coste total**
Mano de obra	25€/h	315h		7875€
COSTES ASOCIADOS				
Concepto	**Coste total**	**Vida útil**	**Uso**	**Imputable**
Papel	5€	-	-	5€
Tinta	7€	-	-	7€
Encuadernación	10€	-	-	10€
CD's	-	-	-	2€
Amortización impresora	400€	6 años	5meses	27,78€
Amortización ordenador	600€	6 años	5meses	41,67€
COSTES ELÉCTRICOS				
Concepto	**Consumo**	**Coste hora**	**Horas**	**Coste total**
Costes eléctricos	250W	0,15€/kWh	315h	11,81€
			TOTAL	7980,26€

Los costes totales ascienden a **7980,26€**.

8. Impacto ambiental

El impacto ambiental del presente proyecto, al no contar con una fase experimental, se reduce a las emisiones de CO_2 derivadas del consumo eléctrico para la realización del mismo y los residuos generados por el uso de consumibles.

Emisiones de CO_2 derivadas del consumo eléctrico

En la Tabla 9.1 se muestra un desglose de las emisiones de CO2 derivadas del consumo eléctrico.

Elemento de consumo	Tiempo de consumo [h]	Consumo de potencia [W]	Consumo de energía [kWh]	Consumo CO_2 [kg][5]
Ordenador	315h	250	78,75	51,19
Impresora	0,75h	150	0,11	0,07
Iluminación	100h	90	9,00	5,85
			TOTAL	57,11

Tabla 9.1. Emisiones de CO_2 derivadas del consumo eléctrico

Residuos producidos por el uso de consumibles

Los consumibles necesarios para la redacción del proyecto son papel en formato A4, tinta para la impresora y diverso material de oficina.

El consumo total de papel se estima en 120 hojas en formato A4, lo que representa unos 7,48m^2 de papel. Con una densidad de 80g/m^2, el peso total utilizado es de 0,60kg.

[5] Según la Comisión Europea, 1kWh produce 0,65kg de CO_2

CONLUSIONES

Aunque el programa se encuentra aún en vías de desarrollo, se han cumplido en gran medida los objetivos planteados. Como ya se ha propuesto en el apartado vías de continuación, existen aún aspectos potenciales de mejora.

La gran heterogeneidad de los datos de partida fue el principal problema que presentó el trabajo durante su desarrollo. Algoritmos clasificatorios, considerados a priori como los más adecuados, tuvieron que ser descartados por este motivo. Aunque las herramientas aplicadas están pensadas para solventar los problemas que la heterogeneidad introduce, condiciona, en mayor o menor grado, los resultados obtenidos.

Como muestran los resultados, el algoritmo *k-means*, junto con el coeficiente *silhouette*, forman una herramienta flexible y eficaz para la caracterización de subgrupos internos en las fuentes.

La introducción de nuevas técnicas de clasificación, como los árboles de decisión y el algoritmo *Random Forest*, permiten una primera aproximación a la predicción de muestras automatizada. Hacer hincapié en los resultados obtenidos a través de los árboles de decisión; una estructura sobreajustada que no otorga un buen modelo de predicción de muestras para el conjunto de datos del estudio. Sin embargo, puede resultar útil para futuros estudios con otros juegos de datos. En contraposición, el algoritmo *Random Forest* consiguió llegar a unos resultados más convincentes, cubriendo así uno de los principales objetivos propuestos. El modelo de validación implementado (*cross validation*) permite cuantificar el grado de certeza de los resultados obtenidos mediante los diferentes modelos de predicción, aspecto fundamental que se marcó como principal vía de continuación.

Las mejoras introducidas en la representación gráfica otorgan una mayor flexibilidad y potencia al software, permitiendo al usuario la representación de cualquier tipo de gráfico. La adición del cálculo de las elipses de error permite un avance en la clasificación de muestras visual que combina con las técnicas de mejores combinaciones aplicadas por el proyectista anterior.

Además, los diferentes algoritmos aplicados se han intentado implementar de la manera más flexible y configurable posible. De tal modo que la herramienta fuese extensible a nuevos juegos de datos y muestras que requiriesen de otras configuraciones.

BIBLIOGRAFIA

Referencias bibliográficas

[1] Howes, Mauricio Andrés Alva. *Implementación y aplicación de herramientas para los estudios de procedencia de objetos arqueológicos de obsidiana.* Barcelona : Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, 2014.

[2] Tepe Y, Analytique LAM. *L'origine des outils en obsidienne de Tell Magzalia, Tell Sotto.*1994; 20: 18-31.

[3] Binder D, Gratuze B, Mouralis D, Balkan-Atlı N. *New investigations of the Göllüdağ obsidian lava flows system: a multi-disciplinary approach.* J Archaeol Sci. 2011; 38(12): 3174-3184.

[4] Frahm E. *Distinguishing Nemrut Dağ and Bingöl A obsidians: geochemical and landscape differences and the archaeological implications* J Archaeol Sci. 2012; 39(5): 1436-1444.

[5] Keller and Seifried. *The present status of obsidian source identification in Anatolia and the Near East.*

[6] Khalidi L, Gratuze B, Boucetta S. *Provenance of Obsidian Excavated From Late Chalcolithic Levels At the Sites of Tell Hamoukar and Tell Brak, Syria.* Archaeometry. 2009; 51(6): 879-893.

[7] Carter T, Shackley MS. *Sourcing Obsidian From Neolithic Çatalhöyük (Turkey) Using Energy Dispersive X-Ray Fluorescence.* Archaeometry. 2007; 49(3): 437-454.

[8] Carter T, Grant S, Kartal M, Coşkun A, Özkaya V. *Networks and Neolithisation: sourcing obsidian from Körtik Tepe (SE Anatolia).* J Archaeol Sci. 2013; 40(1): 556-569.

[9] Documentación oficial Orange

http://docs.orange.biolab.si/reference/rst/index.html.

[10] Coeficiente de Pearson
http://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson.

[11] Coeficiente de Spearman
http://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Spearman.

[12] Duarte, Marcos. Prediction ellipse and prediction ellipsoid
http://nbviewer.ipython.org/github/demotu/BMC/blob/master/notebooks/PredictionEllipseEllipsoid.ipynb.

ANEXO I

El presente manual permite aprender a utilizar todas las funcionalidades de *SourceSeeker 2.0* a cualquier usuario. Está pensado para su utilización en *Windows*, aunque también es compatible con otros sistemas operativos como *Ubuntu*.

Ejecución del programa

SourceSeeker no necesita de la instalación de ningún programa complementario. Sólo es necesario ejecutar el archivo proporcionado en el CD. Tras su ejecución, al cabo de unos segundos, aparecerá por pantalla la interfaz principal del programa (véase Figura 1).

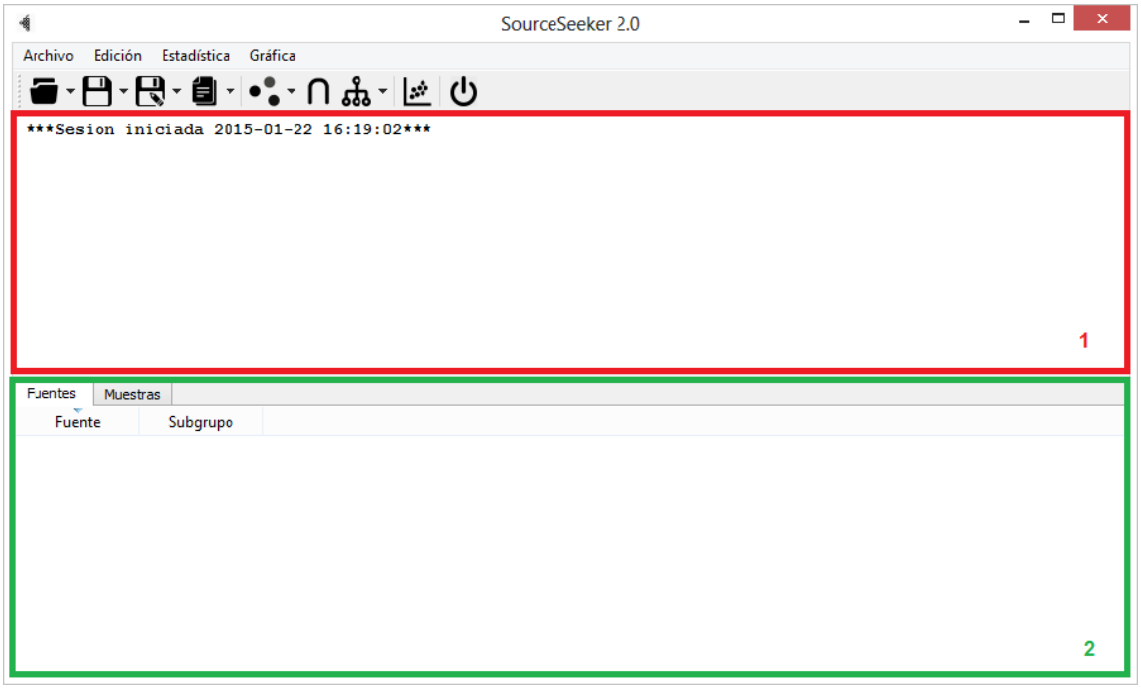


Fig. 1. Interfaz principal del programa

La interfaz principal está dividida en dos partes:

- La parte superior (color rojo) mostrará los resultados de la ejecución de los diferentes algoritmos. Durante el resto de tutorial se hará referencia a esta parte como **hoja de sesión**.
- La parte inferior (color verde) cuenta con dos tablas; una para los datos fuente y otra para los datos muestra. El usuario puede moverse por ellas a través de las pestañas adheridas.

Formato archivos de datos

Los archivos *.xls deben presentar una estructura preestablecida. En la Tabla 1 y Tabla 2 se muestran los esquemas de la estructura para el fichero datos y muestras respectivamente.

Fuente	**Subgrupo**	**Elemento 1 [unidades]**	**Elemento 2 [unidades]**	**...**	**Elemento k [unidades]**
f1	g1	valor 1.1	valor 1.2	...	valor 1.k
...	...	...	...	...	...
fn	gn	valor n.1	valor n.2	...	valor n.k

Tabla 1. Esquema de la estructura de las fuentes de Excel

Muestra	**Elemento 1 [unidades]**	**Elemento 2 [unidades]**	**...**	**Elemento k [unidades]**
m1	valor 1.1	valor 1.2	...	valor 1.k
...	...	...	...	...
mn	valor n.1	valor n.2	...	valor n.k

Tabla 2. Esquema de la estructura de las muestras de Excel

Los valores deben ser numéricos. En caso contrario, el programa impondrá un valor desconocido a estos casos. Además, todas las palabras del archivo se deberán incluir sin acentos u otros caracteres especiales. En contraposición, se pueden producir errores en la ejecución de algunas funcionalidades del programa.

Tratamiento de datos

SourceSeeker permite realizar varias acciones sobre los datos. Todas ellas están disponibles tanto para datos fuente como para datos muestra. Se puede acceder a ellas a través del menú *Archivo* (véase Figura 2).

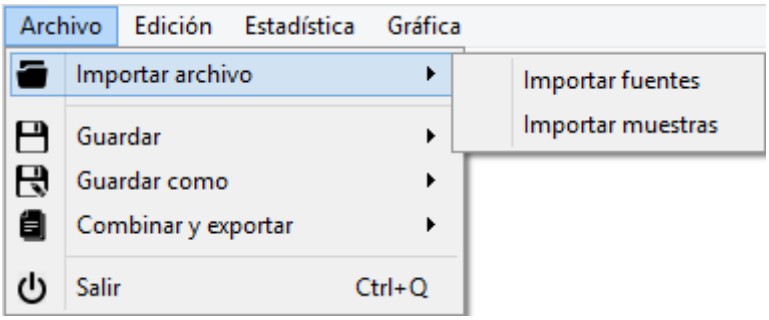


Fig. 2. Menú archivo para la ejecución de tratamiento de datos.

O mediante la barra de herramientas (véase Figura 3)

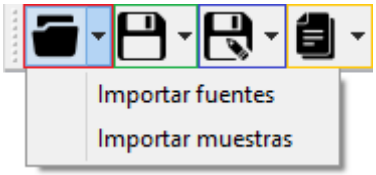


Fig. 3. Barra de herramientas para la ejecución de tratamiento de datos

Se hace una breve explicación del funcionamiento de cada una de las acciones:

- *Importación de ficheros (rojo)*

Importa un fichero a la base de datos del programa. Si previamente existe un fichero importado de la misma categoría, todos los datos son reemplazados.

- *Guardar (verde)*

Guarda las modificaciones realizadas desde el programa en el fichero Excel de origen.

- *Guardar como (azul)*

Guarda los datos, con sus respectivas modificaciones, en un nuevo fichero Excel a indicar por el usuario.

- *Combinar y exportar (amarillo)*

Unifica (siempre que los archivos estén en el formato correcto) el contenido de varios archivos independientes de manera automática. Los archivos que se desean combinar deben estar en la misma carpeta para poder ser seleccionados (Figura 4). En cambio, el nuevo archivo generado puede guardarse en cualquier ubicación (siempre que se le asigne un nombre válido) (Figura 5).

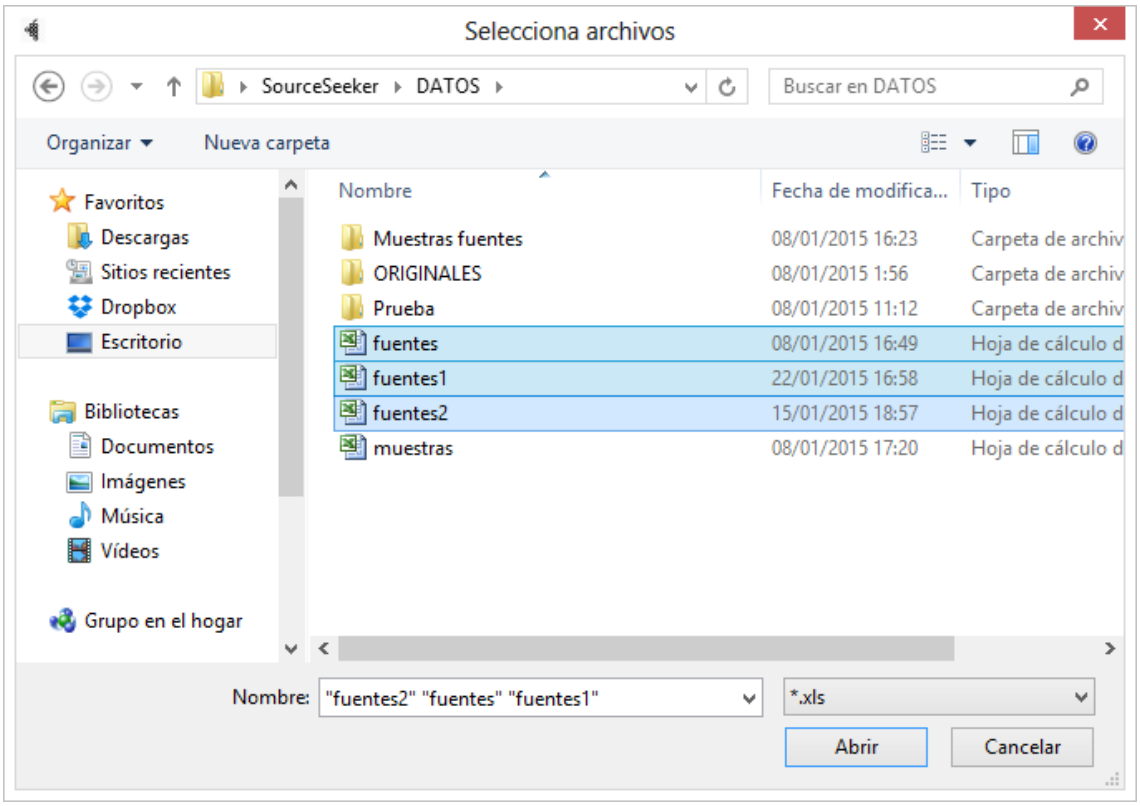


Fig. 4. Ventana de diálogo para la selección de varios archivos *.xls

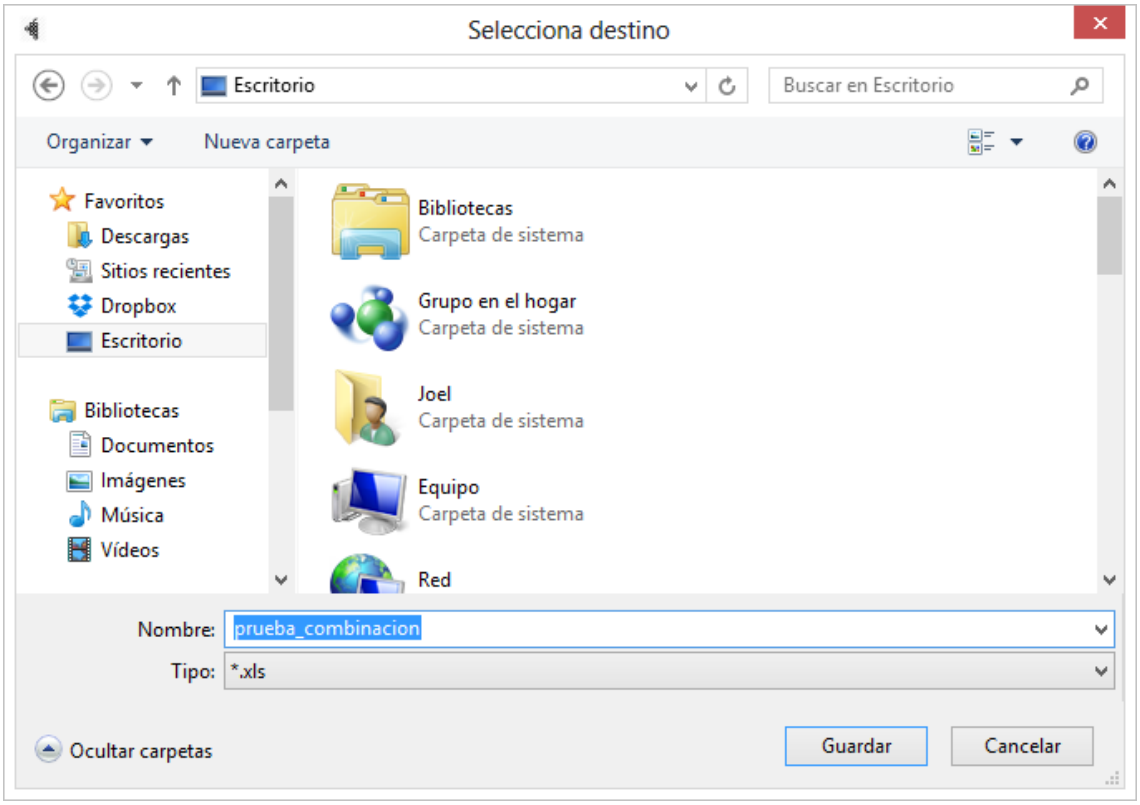


Fig. 5. Ventana de diálogo para la selección del destino y nombre del archivo *.xls

Edición de datos

Ya sea relativo a la hoja de sesión o a las tablas de datos, es posible la eliminación de su contenido mediante herramientas de edición. Se pueden acceder a estas funcionalidades a través del menú *Edición* (véase Figura 6).

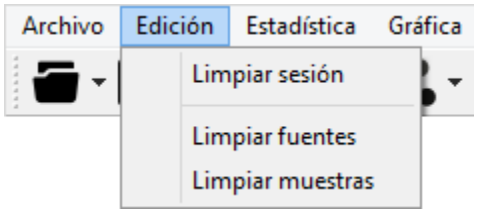


Fig. 6. Menú Edición

Implementación de conglomerados

La implementación de conglomerados contempla los algoritmos *silhouette* y *k-means.* Se pueden acceder a ellos a través de la barra de herramientas (véase Figura 7) y a través del menú *Estadística – Clustering* (véase Figura 8).

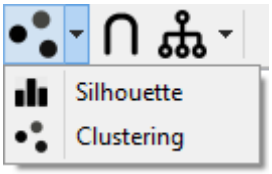


Fig. 7. Barra de herramientas Clustering

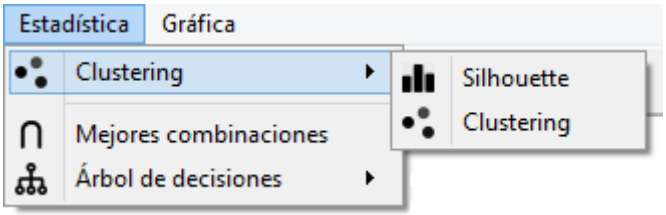


Fig. 8. Menú Estadística - Clustering

En la Figura 9 se muestra el diálogo referente al cálculo del **coeficiente *silhouette***. A través de esta ventana es posible modificar los siguientes parámetros:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.

- *Rango de conglomerados.* Permite seleccionar el rango de conglomerados en el que se quiere trabajar.
- *Semilla.* Permite seleccionar el valor de la semilla.
- *Inicialización.* Permite seleccionar el método de inicialización de los centroides.
- *Distancia.* Permite seleccionar el tipo de distancia con la que se quiere trabajar.
- *Graficar resultados.* Otorga la posibilidad de graficar los resultados de salida.
- *Eliminar compuestos que no explican todas las muestras.* Todo aquel compuesto para el que alguna de la muestras no contenga su valor será ignorado en el cálculo.

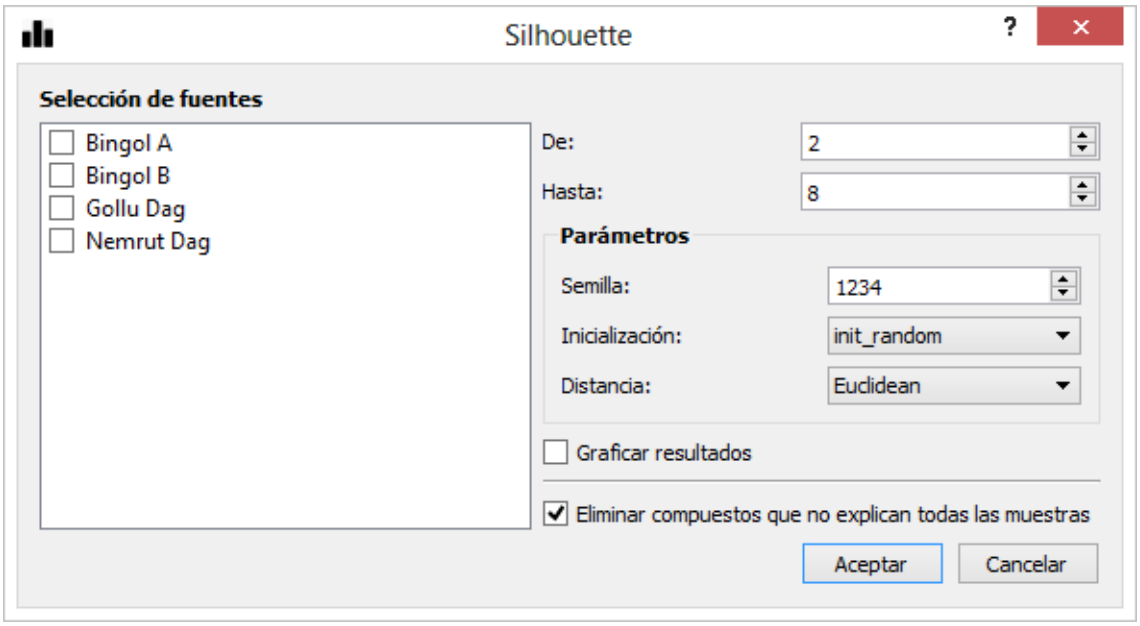


Fig. 9. Diálogo *silhouette*

La implementación del algoritmo **k-means** se realiza a través de la ventana mostrada en la Figura 10. Los posibles parámetros a modificar, muy parecidos al cálculo del coeficiente *silhouette,* son los siguientes:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Número clusters.* Permite seleccionar el número de conglomerados.
- *Semilla.* Permite seleccionar el valor de la semilla.
- *Inicialización.* Permite seleccionar el método de inicialización de los centroides.
- *Distancia.* Permite seleccionar el tipo de distancia con la que se quiere trabajar.
- *Eliminar compuestos que no explican todas las muestras.*

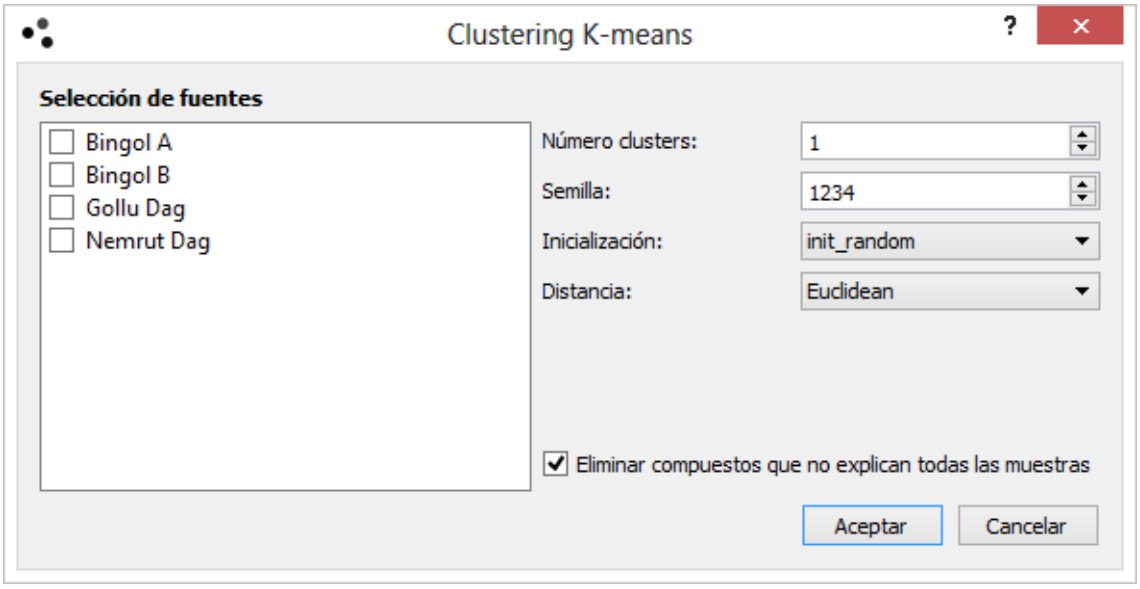


Fig. 10. Diálogo *k-means*

Mejores combinaciones

Las técnicas de **mejores combinaciones** se encuentran disponibles a través de la barra de herramientas (véase Figura 11) y a través del menú *Estadística* (véase Figura 12)

Fig. 11. Barra de herramientas Mejores combinaciones

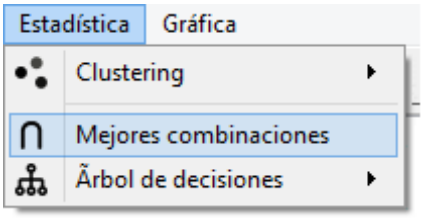


Fig. 12. Menú Estadística - Mejores combinaciones

El diálogo cuenta con los siguientes parámetros a modificar (véase Figura 13):

- *Fuentes obligadas.* Permite seleccionar las fuentes que obligatoriamente entran en juego en el algoritmo.
- *Número mejores.* Permite seleccionar el número de mejores combinaciones que se muestran.
- *Tipo.* Permite seleccionar el tipo de combinaciones con el que se quiere trabajar. Puede ser tanto simple como fraccional.

- *Valores estandarizados.* Permite estandarizar los valores.
- *Graficar resultados.* Permite visualizar en forma de gráfico bidimensional, las mejores combinaciones de salida.

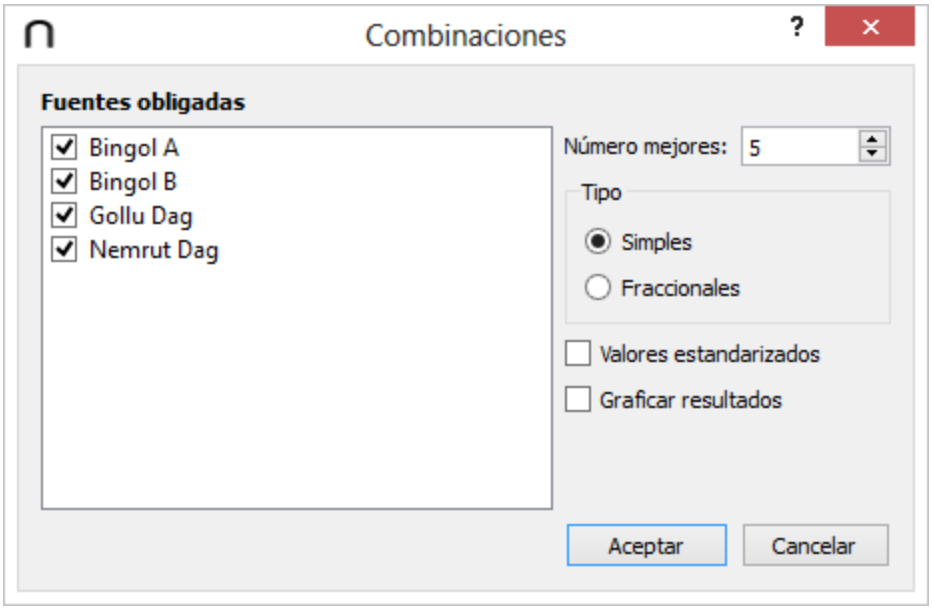


Fig. 13. Diálogo Mejores combinaciones

Implementación algoritmos de clasificación

La implementación de algoritmos de clasificación contempla las técnicas de **árbol de decisión** y ***random forest*** accesibles a través de la barra de herramientas (Figura 14) y a través del menú *Estadística - Árbol de decisiones* (Figura 15).

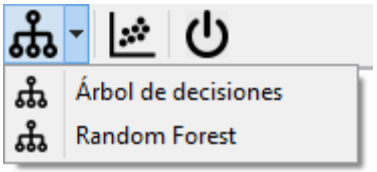


Fig. 14. Barra de herramientas Árbol de decisiones

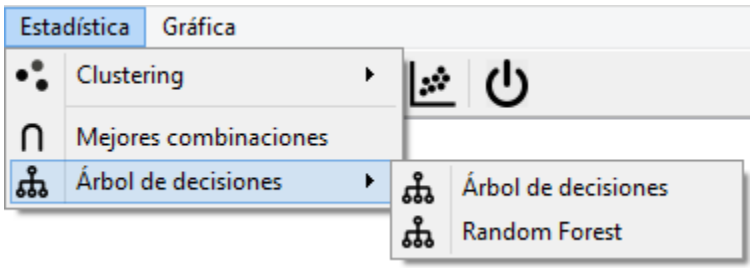


Fig. 15. Menú Estadística - Árbol de decisiones

El diálogo del algoritmo **árbol de decisiones** cuenta con los siguientes parámetros a modificar (Figura 16):

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Selección de muestras.* Permite seleccionar las muestras con las que se quiere trabajar.
- *Splitter*. Permite seleccionar el tipo de *splitter* deseado.
- *Descender.* Permite seleccionar el tipo de *descender* deseado.
- *Profundidad.* Permite limitar la profundidad máxima del árbol resultante.
- *Diferenciar entre subgrupos dentro de una misma fuente.*
- *Mostrar árbol de decisión.*
- *Cross validation.*

Por otra parte, admite indicar el procesamiento de datos deseado.

- Eliminar compuestos que no contiene la muestra.
- Eliminar compuestos que no explican la fuente.

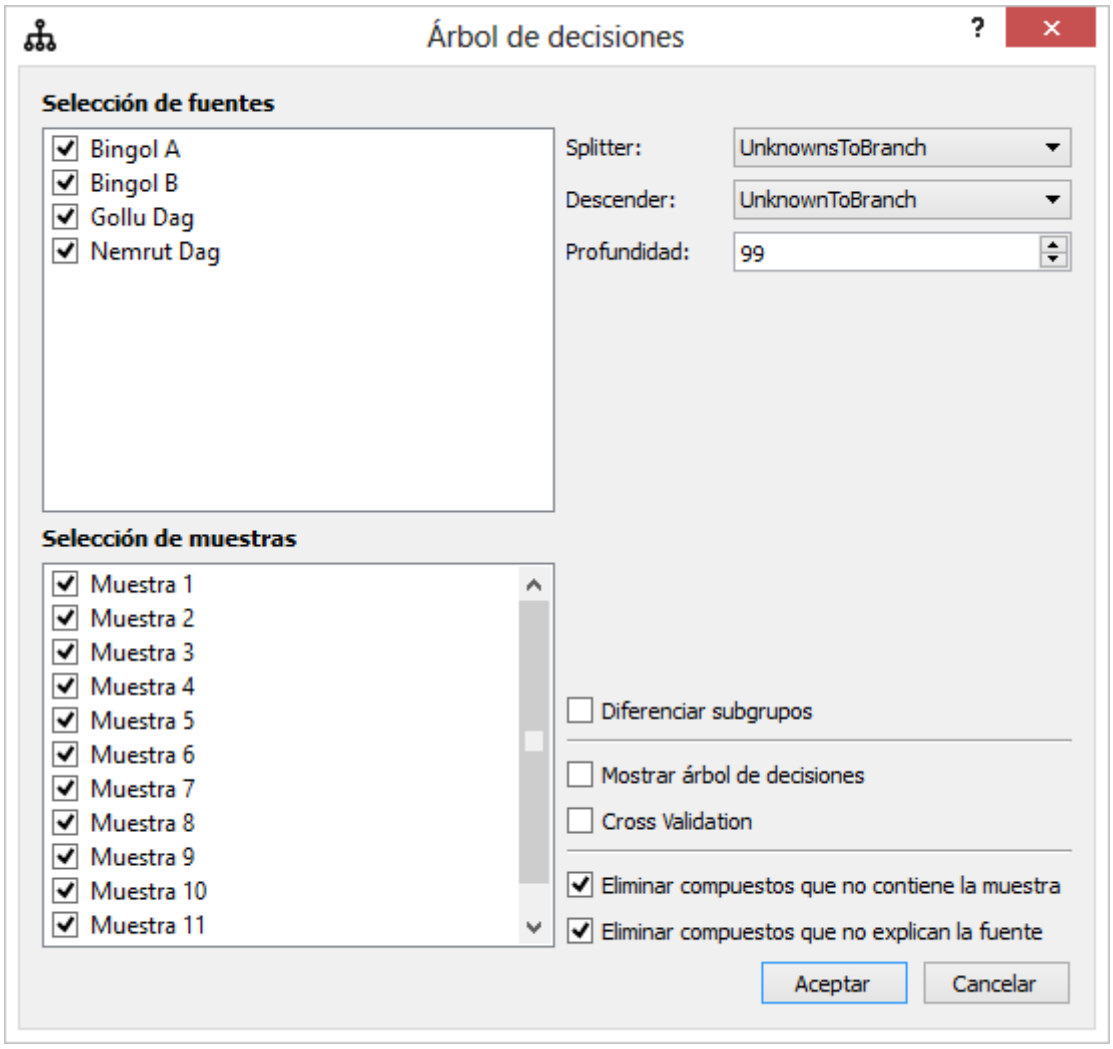


Fig. 16. Diálogo Árbol de decisiones

La implementación del algoritmo ***Random Forest*** se realiza a través de la ventana mostrada en la Figura 17. Los posibles parámetros a modificar son los siguientes:

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Selección de muestras.* Permite seleccionar las muestras con las que se quiere trabajar.
- *Número de árboles.* Permite indicar el número de árboles con el que trabaja el algoritmo.
- *Splitter*. Permite seleccionar el tipo de *splitter* deseado.
- *Descender.* Permite seleccionar el tipo de *descender* deseado.
- *Profundidad.* Permite limitar la profundidad máxima del árbol resultante.
- *Diferenciar entre subgrupos dentro de una misma fuente*.
- *Cross validation.*

Admite indicar también el procesamiento de datos deseado:

- Eliminar compuestos que no contiene la muestra.
- Eliminar compuestos que no explican la fuente.

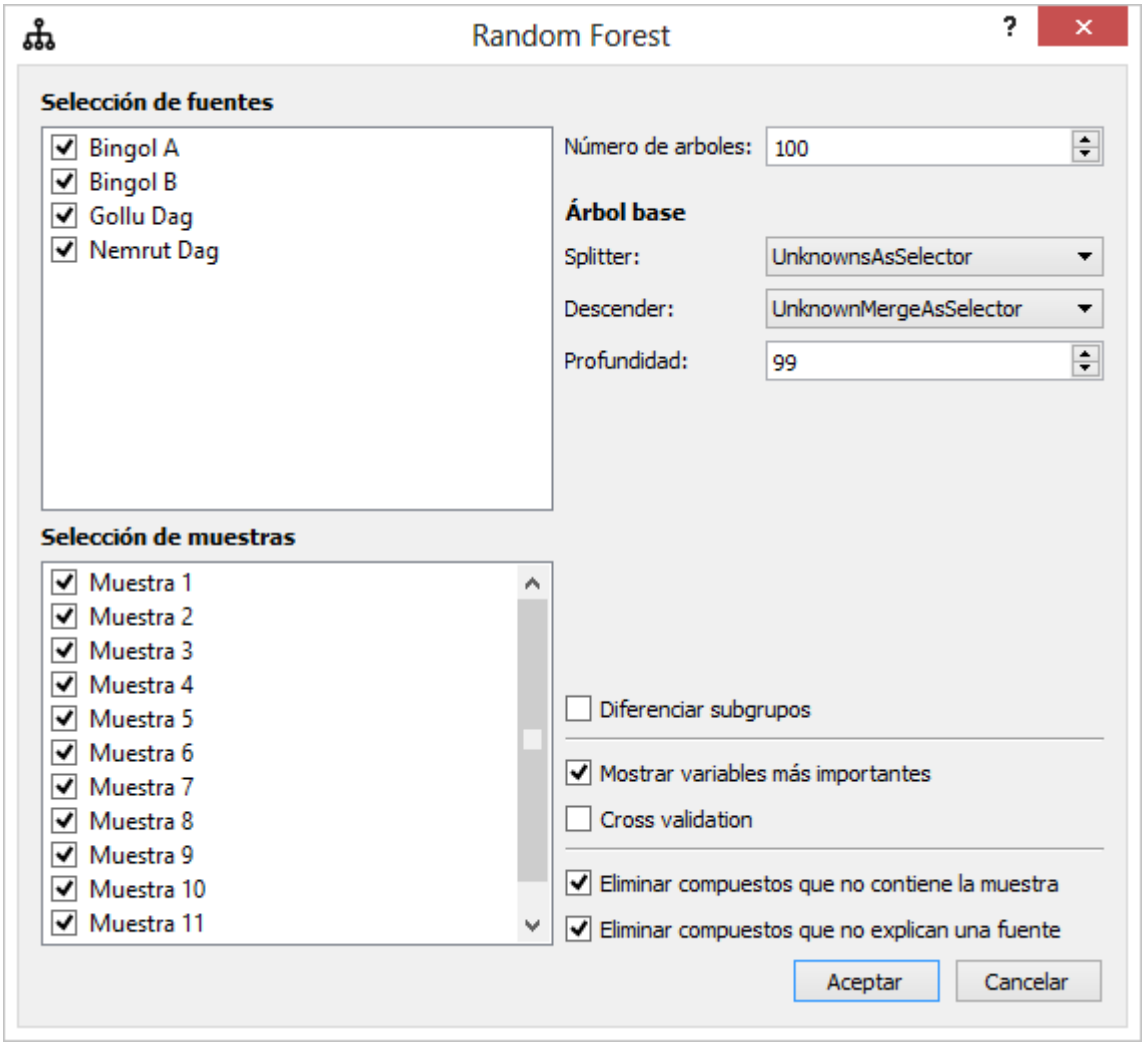


Fig. 17. Diálogo *Random Forest*

Representación gráfica

Se puede acceder a la funcionalidad de representación gráfica a través de la barra de herramientas (véase Figura 18) y a través del menú *Gráfica* (Figura 19).

Fig. 18. Barra de herramienta - Gráfica

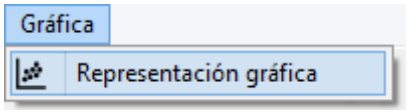


Fig. 19. Menú Gráfica

La ventana final de aplicación presenta los siguientes parámetros a modificar (véase Figura 20):

- *Selección de fuentes.* Permite seleccionar las fuentes con las que se quiere trabajar.
- *Selección de muestras* .Permite seleccionar las muestras con las que se quiere trabajar.
- *Introducción de la funciones en cada uno de las coordenadas.*[6]
- *Diferenciar subgrupos.*
- *Generar elipse de error y p-valor.* Dado un p-valor, permite el cálculo y representación de la elipse de error. Sólo disponible en representaciones bidimensionales.

[6] Recordar que la introducción de funciones debe ser compatible con la librería *math* de *Python*.

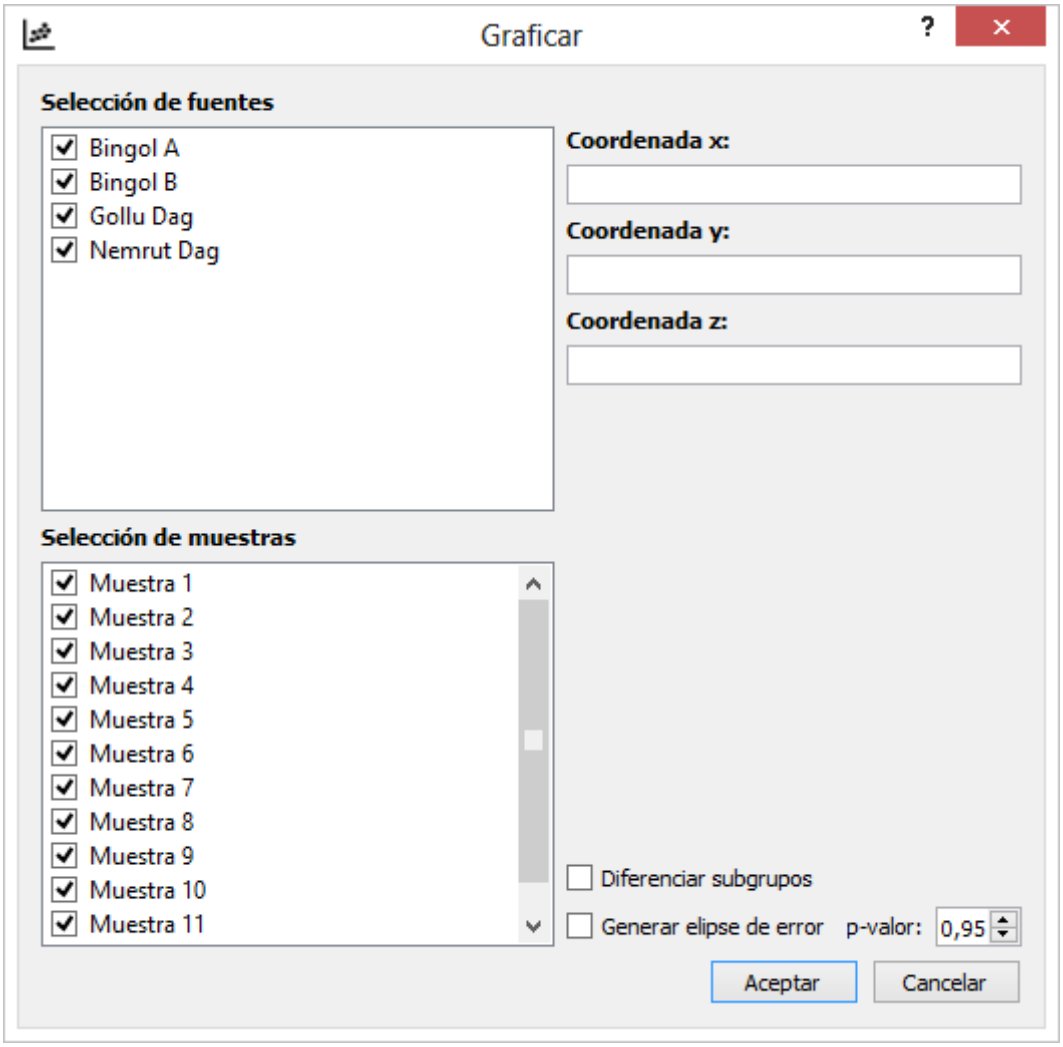


Fig. 20. Diálogo *Random Forest*

Cabe destacar que los compuestos deben introducirse envueltos de los caracteres '<>'. La función de autocompletado entra en funcionamiento al introducir el carácter '<'. La detección de gráficos bidimensionales o tridimensionales se realiza automáticamente al introducir o eliminar una función de la *coordenada z*.

Consejos de uso

1. El fichero de importación debe ser *.xls y no *.xlsx. Si se dispone de una versión de Excel superior a la de 2003, cambiar las opciones al guardar.
2. A la hora de combinar archivos, vigilar que los encabezados estén bien escritos y, en el caso de haber algún elemento común, que esté en las mismas unidades.
3. Para el mismo caso, asegurarse que los archivos a combinar están en la misma carpeta.
4. Conviene dedicar unos minutos a la comparación de los datos, ya que se podría dar el caso que, para el estudio químico, se diesen los mismos datos expresados en otras unidades y haga falta requerir de alguna transformación para poder ser comparados.
5. No se recomienda el análisis simultáneo de muestras que tengan compuestos muy dispares entre ellos. Esto puede producir una reducción drástica del número de compuestos con los que trabaja el algoritmo.
6. Se recomiendo aplicar siempre filtros de eliminación de compuestos. En caso contrario, el algoritmo trabaja con muchas variables (compuestos) y los tiempos de ejecución pueden ser muy elevados, llegando hasta los 5 minutos.

ANEXO II

La presente documentación ofrece una primera aproximación al funcionamiento interno de *SourceSeeker*. Para ello se realiza una breve descripción a nivel global de la estructura del programa.

Cuenta con los siguientes archivos:

UI_SourceSeeker.py

Se encuentra la construcción de la interfaz de cada una de las ventanas del programa mediante la definición de las siguientes clases:

- *UI_SourSeeker*
- *UI_Silhouette*
- *UI_Clustering*
- *UI_Combinaciones*
- *UI_ArbolDesciones*
- *UI_RandomForest*
- *UI_Graficar*

SourceSeeker.py

Posee los editores/controladores de cada una de las clases introducidas en el archivo anterior. Contiene las siguientes clases:

- *SourceSeeker*
- *EditorSilhouette*
- *EditorClustering*
- *EditorCombinaciones*
- *EditorArbolDecisiones*
- *EditorRandomForest*
- *Graficar*

Todas ellas heredan de *QtGui.QDialog*, excepto la clase SourceSeeker que lo hace de *QMainWindow.*

datos.py

Contiene todas las funciones y clases que conllevan el tratamiento de datos del programa:

- *Medición* (clase)

Almacena los valores de una medición en un diccionario tipo {compuesto: valor}. El valor del compuesto puede ser tanto un valor (un número) como un valor perdido (None).

- *Fuente* (clase)

Hereda de la clase Medición. Representa una medición fuente. Contiene:

1. Fuente.
2. Subgrupo.
3. Diccionario de variables.

- *Muestra* (clase)

Hereda de la clase Medición. Representa una medición muestra. Contiene:

1. Muestra.
2. Diccionario de variables.

- *Data* (clase)

Contenedor de objetos Medicion. Parte de un archivo Excel donde se encuentran los datos. Esta clase fue concebida con la idea de poder seleccionar y extraer la información deseada en cada momento.

- *DataFuente* (clase)

Contenedor de objetos Fuente. Hereda de la clase Data. Incluye algunas funcionalidades especiales referentes al almacenaje de fuentes.

- *DataMuestra* (clase)

Contenedor de objetos Muestra. Hereda de la clase Data. Incluye algunas funcionalidades especiales referentes al almacenaje de muestras.

- *TablaModelo* (clase)

Genera el modelo de datos necesario para la vista *QTableView*.

- *Formula* (clase)

Objeto para el cálculo de funciones provenientes de la representación de gráficos.

combinaciones.py

Se encuentran definidas todas las funciones y clases referentes al cálculo interno de los diferentes parámetros que necesita la *PseudoF.* Para más información consultar ANEXO II de la memoria (1).

estadistica.py

Se llevan a cabo todos los cálculos estadísticos del programa. Depende principalmente de la librería externa *Orange* y el módulo combinaciones. El cálculo se realiza a partir de los datos fuentes y/o datos muestra y de una serie de parámetros de configuración propios de cada algoritmo.

- *Silhouette*
- *Clustering*
- *Combinaciones*
- *ArbolDecisiones*
- *RandomForest*

elipse.py

Permite el cálculo de la elipse de error a partir de dos listas de datos (valores coordenada x e y) y el p-valor deseado. Se apoya fundamentalmente en las librerías *Numpy* y *Scipy.*

importa.py

Contiene todas las funciones referentes al importado o exportado de datos de ficheros Excel.

- *importar_fuentes* (función)

A partir de un archivo Excel con una distribución preestablecida (estructura fuente), crea y almacena objetos Medicion. Retorna lista de objetos Medicion.

- *importar_muestras* (función)

Muy similar a la función anterior, pero trabaja con archivos con una distribución muestra.

- *exportar_datos* (función)

Dado un dominio y una matriz, exporta los datos a un fichero Excel indicado previamente en los parámetros de entrada.

- *union_datos* (función)

A partir de una lista de objetos Data y el nombre del archivo de destino combina los objetos Data y los guarda en el archivo indicado.

graficas.py

Contiene todas las funciones y clases que hacen referencia a la representación gráfica de datos. Se basa en el uso de la librería *Matplotlib.*

- *Scatterplot* (clase)

Hereda de la clase *QtGui.QDialog*. Facilita la obtención y representación del lienzo donde posteriormente se dibujan los puntos deseados.

- *scatterplot* (función)

A partir del objeto *Scatterplot*, dibuja y muestra por pantalla el gráfico. Se apoya en el uso de un iterador de colores y *marker*s y funciones externas que facilitan la extracción de datos.

Por último, destacar los módulos empleados para su implementación:

- *xlrd* (https://pypi.python.org/pypi/xlrd)
- *xlwt* (https://pypi.python.org/pypi/xlwt)
- *PyQt4* (https://wiki.python.org/moin/PyQt4)
- *Numpy* (http://www.numpy.org/)
- *Scipy* (http://www.scipy.org/)
- *Matplotlib* (http://matplotlib.org/)
- *Orange* (http://orange.biolab.si/)