UNIVERSIDAD DE GRANADA

MÁSTER EN ESTADÍSTICA APLICADA

TRABAJO FIN DE MÁSTER

Software disponible en técnicas de minería de datos. Comparativa y aplicación a datos reales.

Realizado por:

Alberto Díaz Ribón

Tutora:

Rocío Raya Miranda

Curso académico 2021/2022

Índice

1. RESUMEN

En las últimas décadas, tanto en el ámbito de la investigación como en el empresarial, la ingente cantidad de datos de la que se dispone y que se genera cada segundo, demanda tanto nuevas tecnologías de almacenamiento como herramientas y técnicas para su análisis con la finalidad de extraer conocimiento de estos.

Debido a este auge, van surgiendo numerosos softwares gratuitos (creados principalmente en universidades y entornos académicos) para aplicar esas técnicas de análisis, con una comunidad de usuarios o empresas (que acaban adquiriendo el software para su explotación) que fomentan su mejora continua, creando contenidos para su aplicación y aprendizaje. Pese a tener funcionalidades de pago, principalmente enfocados al mundo empresarial, su versión gratuita ya cuenta con una potencia suficiente para aplicar las técnicas y realizar estudios con datos estáticos, es decir, que no se alimenten en tiempo real de forma *online*.

Este trabajo se centra en cuatro de estos softwares gratuitos, ya que la lista es muy amplia y en constante crecimiento. Se han escogido los que se consideran mejor valorados tras consultar estudios como el de Altalhi *et al.* (2017). Para ello, se han elegidos dos de las técnicas más representativas de la minería de datos y, tras realizar una introducción teórica, se aplicarán de forma práctica en cada uno de los programas para analizar dos conjuntos de datos:

- **Árboles de decisión**, usuales en técnicas de aprendizaje supervisado, para predecir, en función de una serie de variables, si una reserva hotelera va a ser cancelada o no.
- Agrupamiento o *clustering* con un algoritmo llamado ***k-means***, asociado a las técnicas de aprendizaje no supervisado, para agrupar un conjunto de vinos en tres grupos según sus características.

Para finalizar, el documento concluye con una comparativa entre los softwares, tanto de una forma cuantitativa como cualitativa, según los resultados obtenidos tras la aplicación de las técnicas y la experiencia de uso, reflexionando sobre cuál de ellos ofrece mejores resultados en ambos sentidos.

2. INTRODUCCIÓN: ¿QUÉ ES LA MINERÍA DE DATOS?

La **minería de datos** es un conjunto de técnicas que, mediante el estudio de grandes volúmenes de datos, tiene como objetivo la extracción de conocimiento a partir de los mismos.

Pese a ser un concepto nacido en la década de los 90, el auge de los últimos años viene dado por una serie de acontecimientos que han coincidido en el tiempo, como son el abaratamiento de la tecnología, tanto en lo que afecta al almacenamiento como al procesamiento de los datos, y el incremento en la velocidad de procesamiento y de transmisión de los datos, acentuándose aún más con los recientes avances en computación en la nube y *big-data*.

Esa aplicación de técnicas y tecnologías se deben integrar en un proceso iterativo en el que tampoco se debe olvidar la comprensión del negocio (o ámbito) en el que se aplique, pudiéndose resumir en seis fases: Comprensión del negocio, comprensión de los datos, preparación de los datos, modelado de los datos, evaluación de los resultados y despliegue o puesta en producción del modelo, según Shearer (2000) en su modelo CRISP-DM (Cross Industry Standard Process).

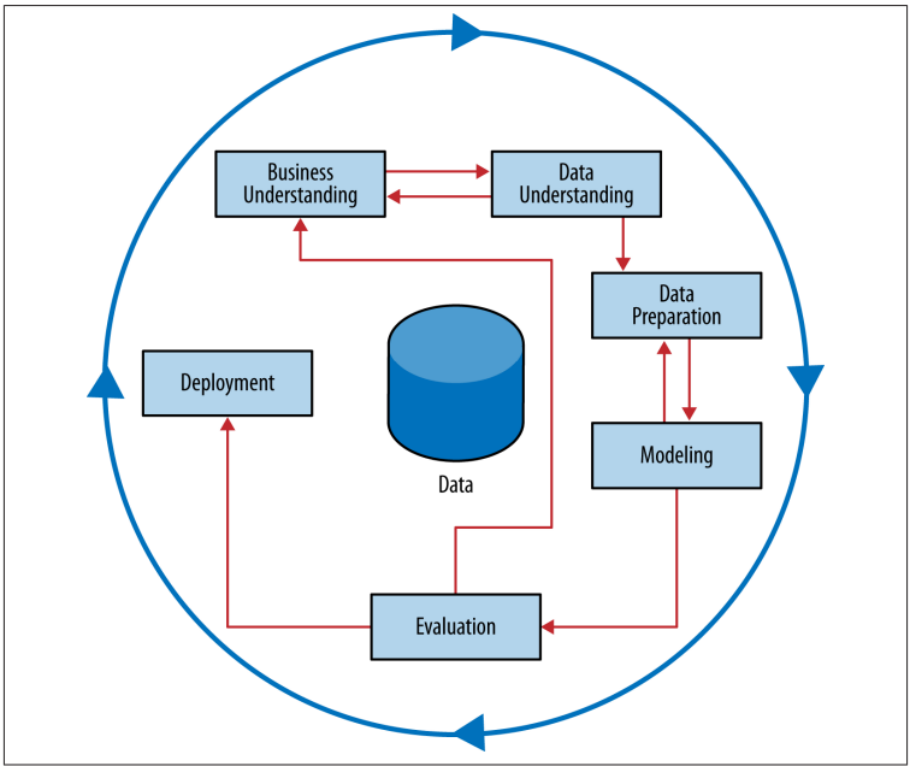


Figura 1. Proceso de minería de datos. Modelo CRISP-DM (Fuente: Provost *et al.,* 2013)

Como indican Riquelme Santos *et al.* (2006), las tareas de la minería de datos pueden ser:

- Descriptivas, describiendo patrones o relaciones en los datos disponibles.

- Predictivas, como a la hora de clasificar nuevos datos en función de los disponibles a modo de histórico.

División que actualmente se relaciona con los dos grandes grupos de técnicas de minería de datos: técnicas supervisadas (asociadas a tareas predictivas) y no supervisadas (relacionadas con las tareas descriptivas).

3. TÉCNICAS SUPERVISADAS

Las técnicas de aprendizaje supervisado se caracterizan por tener una variable objetivo, por lo que la finalidad será predecir esa variable en función del resto de atributos del conjunto de datos.

Si el objetivo es una variable nominal (categórica) también se suele denominar *clase*, ya que la finalidad del modelo será clasificar las instancias en cada una de esas clases preestablecidas, mientras que, si la variable es cuantitativa, lo más común es aplicar técnicas de predicción/regresión.

Teniendo en cuenta el tipo de variable objetivo, las técnicas supervisadas más comunes se pueden dividir de la siguiente forma:

- Clasificación:
  - Tabla de decisión
  - Árboles de decisión
  - Inducción de reglas
  - Clasificación Bayesiana
  - Basada en ejemplares
  - Redes neuronales
- Predicción:
  - Regresión
  - Árboles de predicción
  - Estimador de núcleos

A continuación, se amplía la información sobre el funcionamiento de los árboles de decisión, al ser el algoritmo que, posteriormente, se aplicará de forma práctica con todos los softwares estudiados y comparados.

3.1. Clasificación (árboles de decisión)

Un **árbol de decisión** es una técnica de modelado predictivo que puede ser utilizado tanto para tareas de clasificación como de regresión, según si la variable objetivo es discreta o continua, respectivamente.

Una forma simple de entender esta técnica es ver el árbol como una serie de condiciones organizadas en una forma estructurada y jerárquica (en

forma de árbol), con diferentes nodos interconectados con ramas, pudiendo diferenciar:

- **Nodo raíz**: Nodo inicial del que parten las primeras ramas del árbol, normalmente representado en la parte superior del mismo, en el que está contenido todo el conjunto de datos.
- **Nodo intermedio**: Nodo que recibe tanto ramas entrantes (*inputs*) como salientes (*outputs*). Cada nodo tiene como finalidad dividir el conjunto de datos que recibe como *input* en base a una pregunta lógica (a modo de función de decisión).
- **Hoja** (nodo terminal): Sólo tiene ramas entrantes y se asocia con un valor o etiqueta para los datos que llegan a esta hoja, siendo la partición final, representando de esta manera la decisión final de las instancias que lleguen a esta partición final, entendiéndose como el valor esperado de la variable objetivo.

Por tanto, una vez creado el modelo, la clasificación de una nueva instancia recorrería un camino desde el nodo raíz hasta una de las hojas "viajando" por cada uno de los nodos intermedios en función de cada respuesta a las preguntas que recibe sobre sus atributos en esos nodos intermedios.

La construcción del árbol se basa en dividir el conjunto de datos en subconjuntos cada vez más pequeños hasta llegar a una hoja en la que se considere que sea lo suficientemente "pura", estableciendo para ello una serie de condiciones de parada en la construcción del árbol, siendo las más comunes:

- **Pureza del nodo**: Se suele establecer una proporción máxima de pureza por nodo ya que, en el caso de no hacerlo, el árbol construiría nodos que solamente contengan instancias de una sola clase. La idea consiste en determinar un nivel de pureza suficiente para obtener buenos resultados sin llegar a formar un árbol demasiado complejo.
- **Cota de profundidad**: También para evitar la formación de árboles excesivamente complejos y profundos, esta cota representa el umbral máximo de niveles que puede existir en el árbol.
- **Umbral del soporte**: También se puede definir un número mínimo de instancias por nodo, deteniendo la construcción del modelo cuando en un nodo se considera que no hay suficientes instancias para obtener una nueva regla de división.

El establecimiento de estas reglas es un proceso de configuración muy importante para evitar la construcción de árboles con un exceso de complejidad que se aprendan tan bien las características del conjunto de datos que, a la hora de recibir nuevas instancias que clasificar con características distintas, no sea capaz de clasificarlos con un mínimo nivel de acierto. Esto es conocido como "sobreajuste" y provoca que el modelo de clasificación no sea capaz de generalizar, es decir, de descubrir qué relación existe entre las variables independientes y la dependiente.

Otro aspecto fundamental para tener en cuenta es que la clasificación de cada instancia tendrá una probabilidad de pertenecer a la clase positiva, por lo que habrá que elegir un umbral a partir del cual se considerará una instancia clasificada como positiva, siendo negativa en el caso de que sea inferior a ese umbral.

3.2. Métricas de evaluación de un clasificador

Se detallan algunas de las métricas utilizadas para evaluar el rendimiento de un clasificador binario, que posteriormente se utilizarán en los diversos softwares.

- **Matriz de confusión**:

Según las predicciones que realiza el modelo y los valores reales de cada instancia, se construye la siguiente matriz para, posteriormente, poder calcular el resto de las métricas:

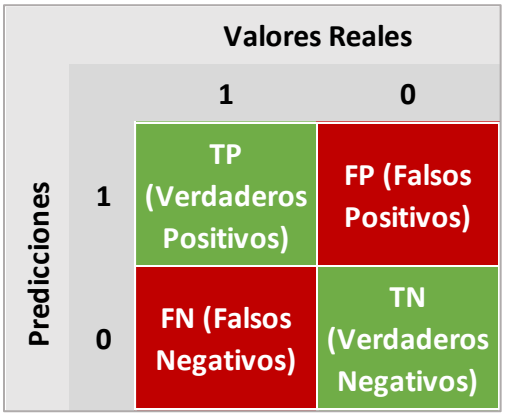


Figura 2: Matriz de confusión (Elaboración propia).

- **Exactitud (*accuracy*)**:

La métrica más genérica en los clasificadores, midiendo la capacidad de clasificar correctamente las instancias, independientemente de la clase a la que pertenezcan.

$$Exactitud = \frac{\text{Instancias bien clasificadas}}{\text{Instancias totales}} = \frac{\text{TP} + \text{TN}}{\text{Instancias totales}}$$

Hay que tener en cuenta que esta métrica no es la más adecuada cuando las clases están desbalanceadas debido a que, como señalan Thabtah *et al.* (2020), en un conjunto de datos con una clase muy desbalanceada se corre el riesgo de que el modelo acabe clasificando gran parte de las instancias en esa categoría mayoritaria para lograr un buen resultado en exactitud.

- **Sensibilidad** (o *true positive rate*):

Sirve para conocer la capacidad del clasificador para detectar los casos positivos sobre los que son realmente positivos.

$$Sensibilidad = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- **Especificidad** (o *true negative rate*):

Su finalidad es conocer la capacidad del clasificador para detectar los casos negativos sobre los que son realmente negativos.

$$Especificidad = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

- **Curva ROC** (*Receiver operating characteristic*):

Es una curva que permite evaluar la capacidad del modelo para clasificar correctamente, relacionando las dos medidas anteriores: la especificidad en el eje *X* y la sensibilidad en el eje *Y*. Esas dos medidas se pueden variar en función del umbral que se decida tomar como predicción de un valor positivo, es decir, la probabilidad a partir de la cual una predicción se considera clasificada como positiva. Cada punto de la curva ROC representa cómo se comporta el clasificador según todos los posibles umbrales.

Como indican Fawcett y Provost (1997), evaluar un clasificador mediante la representación de esta curva tiene la ventaja de observar de forma rápida:

- Un rendimiento óptimo cuanto más se acerque la curva a la esquina superior izquierda.
- Un rendimiento similar al clasificar las instancias al azar si se iguala con la diagonal.
- Un rendimiento peor a clasificar las instancias al azar si se acerca a la esquina inferior derecha.

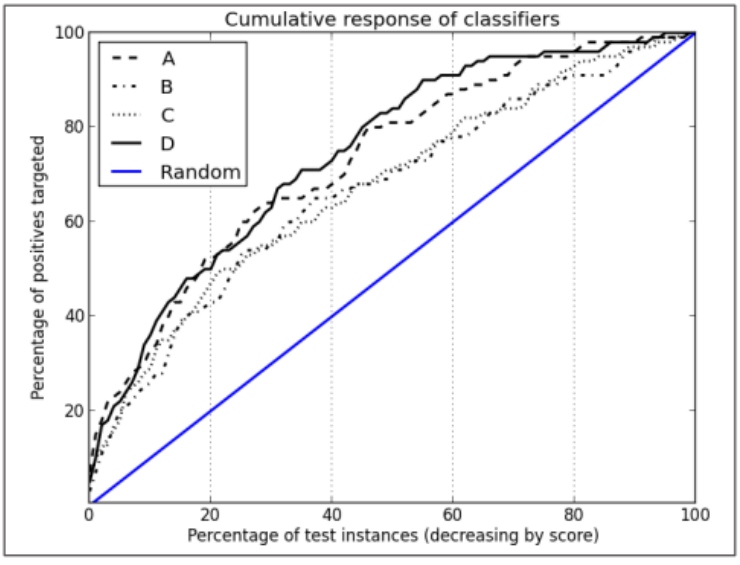


Figura 3. Ejemplo de diferentes curvas ROC (Fuente: Provost *et al.*, 2013)

- **Área AUC** (*Area Under the Curve*):

Es el área bajo la curva ROC, una métrica para conocer qué curva ROC es mejor, ya que en ocasiones puede ser difícil apreciarlo a simple vista. Obviamente, cuanto más se acerque a 1 mejor será el clasificador.

- **Validación cruzada** (*Cross-Validation*):

Pese a no tratarse de una métrica en sí misma, es una técnica de validación muy utilizada, sobre todo cuando se cuenta con una muestra de datos no demasiado grande, que consiste en que el conjunto de entrenamiento (con el que se crea el modelo) se divide en *k* conjuntos más pequeños Para cada uno de los *k* "pliegues" (*folds* en inglés) se sigue el siguiente procedimiento:

  - Un modelo se entrena usando *k* – 1 de los pliegues como datos de entrenamiento
  - El modelo resultante se valida con la parte restante de los datos (es decir, se utiliza como conjunto de test para calcular una medida de rendimiento como, por ejemplo, la exactitud).

4. TÉCNICAS NO SUPERVISADAS

Las técnicas no supervisadas se diferencian de las supervisadas en que, en esta ocasión, no se dispone de una variable objetivo, motivo por el que se trata de un aprendizaje "sin supervisión".

Desde este enfoque, se trata de agrupar u ordenar todas las instancias en una jerarquía según las asociaciones que existan entre todos los atributos de cada una de las observaciones, sin que exista un atributo especial de clase, sin que exista una "etiqueta".

Las técnicas no supervisadas se pueden dividir en:

- ***Clustering***
  - Numérico
  - Conceptual
  - Probabilístico
- **Asociación**
  - A priori

El trabajo se centrará en la aplicación de uno de los algoritmos más utilizados para tareas de agrupación (*clustering*) como es el *k-means,* cuyo funcionamiento se procede a detallar en el siguiente apartado por ser la técnica que se aplicará en los cuatro softwares seleccionados.

4.1. Clustering (K-means)

Es un método de agrupación o clasificación no supervisado que agrupa instancias en *k* grupos teniendo en cuenta sus características, minimizando la suma de una distancia seleccionada con antelación, que suele ser la distancia euclídea, como señalan Ramírez *et al.*, (2003).

El procedimiento iterativo del algoritmo se detalla a continuación:

1. Elegir el número de *clusters k*, es decir, el número de particiones o agrupaciones en la que se desea segmentar el conjunto de datos.
2. Inicializar las coordenadas de cada uno de los centroides, siendo éstas aleatorias.
3. Se calcula la distancia elegida de cada punto a cada centroide y dicho punto se asigna al centroide más próximo según esa medida.

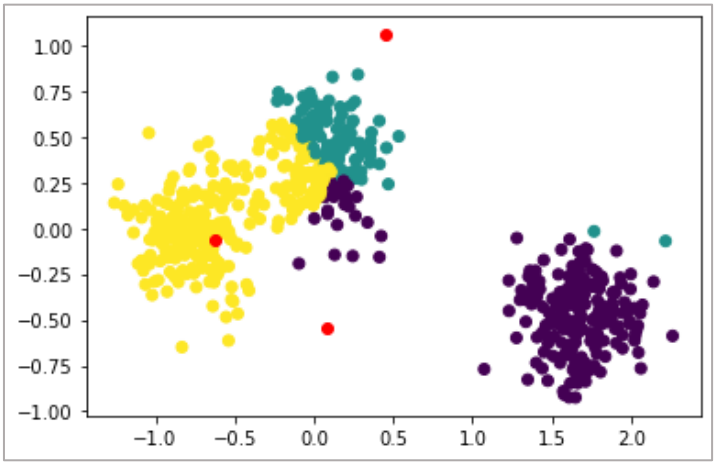


Figura 4. Paso 3 del algoritmo de *k-means* (Elaboración propia).

4. Una vez están todos los puntos asignados a un clúster, se recalculan los nuevos centroides de forma que vuelven a ser los centros de cada cluster.

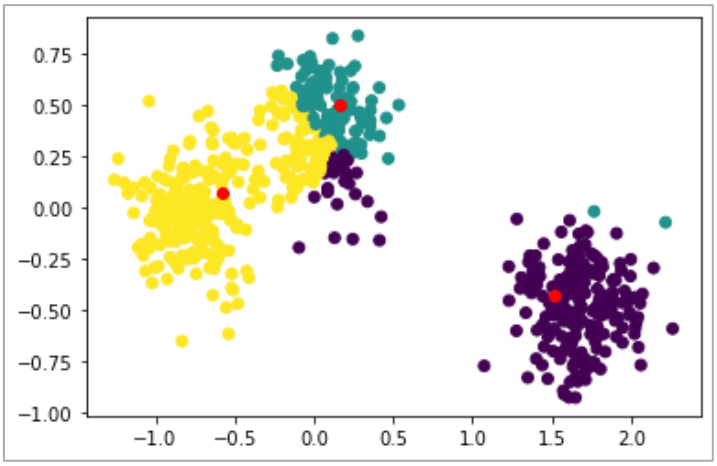


Figura 5. Paso 4 del algoritmo de *k-means* (Elaboración propia)

5. Se realiza un proceso iterativo en los puntos 3 y 4 hasta que se alcanza uno de los criterios de parada.

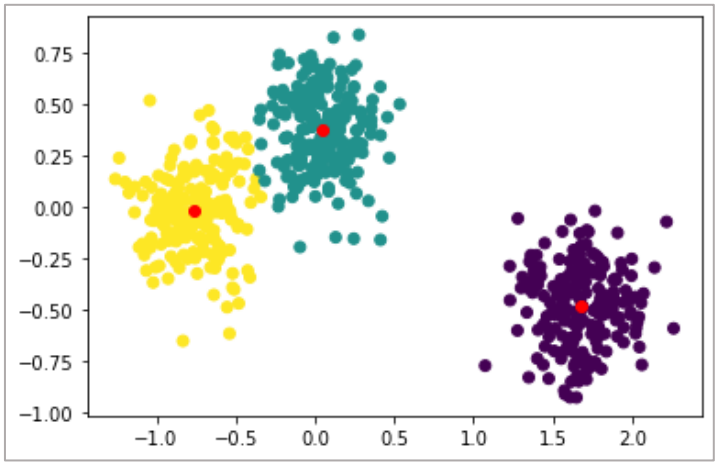


Figura 6. Paso 5 del algoritmo de *k-means* (Elaboración propia)

Existen 3 criterios para terminar el algoritmo:

1. Tras múltiples iteraciones, los centroides de cada *cluster* dejan de cambiar, entendiendo que el algoritmo converge.
2. Tras múltiples iteraciones, los puntos que componen el conjunto de datos dejan de cambiar de *cluster*, entendiendo que el algoritmo converge.
3. Termina el límite de iteraciones prefijado. Debería ser suficiente para considerar que el agrupamiento no va a mejorar de forma sustancial o en aquellas situaciones en las que se pretende evitar un gasto computacional excesivo en datos con un elevado número de observaciones.

Tipos de distancia: Algunas de las distancias más habituales entre 2 puntos o instancias x e y de dimensión n, son las siguientes:

- **Distancia Euclídea**. La distancia más habitual e intuitiva, siendo la longitud de la recta que une dos puntos en el espacio euclidiano:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Distancia de Manhattan o distancia de bloque o ciudad.** Se refiere a recorrer una distancia entre 2 puntos como se haría en la ciudad de Manhattan, es decir, en *zigzag*:

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Distancia de Chebyshev o del máximo**. Se calcula la mayor de las diferencias entre sus dimensiones. También se llama la distancia del tablero de ajedrez, ya que se puede interpretar como el número mínimo de movimientos que la figura del rey necesita para moverse de una casilla a otra:

$$d(x,y) = \max_{i=1..n} |x_i - y_i|$$

- **Distancia de Mahalanobis**. En esta ocasión se utiliza la matriz de covarianzas S:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

4.2. Métricas de evaluación en *clustering* (métrica *Silhouette*):

La métrica de silueta (o *silhouette*) es una de las métricas más comunes en los softwares de minería de datos para determinar el número óptimo de *clusters*. La puntuación de silueta es una medida de cómo de similar es un objeto a su propio grupo en comparación con otros grupos. La puntuación de silueta cercana a 1 indica que la instancia de datos está cerca del centro del clúster y las instancias que poseen puntuaciones de silueta cercanas a 0 están en el límite entre dos clústeres.

Para calcularlo, dado un punto cualquiera del conjunto de datos, se define:

- Cómo de bien el punto está unido al resto de puntos de su *cluster* $a(i) =$ el promedio de la distancia del punto (i) a cada uno de los otros puntos del mismo *cluster* (cuanto menor sea el valor, mejor ha sido asignado el punto)

- Cómo de bien está el punto separado de los puntos del *cluster* vecino más cercano $b(i) =$ la menor distancia promedio del punto (i) a cualquier otro cluster que no contenga (i) (dicho *cluster* es el vecino más cercano de (i) ya que es la mejor alternativa de clasificación del punto)

$$S(i) = \frac{b(i) - a(i)}{\mathrm{m}\ \{a(i), b(i)\}}$$

$$S(i) = \begin{cases} 1 - \dfrac{a(i)}{b(i)} & \text{si } a(i) < b(i) \\ 0 & \text{si } a(i) = b(i) \\ \dfrac{b(i)}{a(i)} - 1 & \text{si } a(i) > b(i) \end{cases}$$

$$-1 \leq S(i) \leq 1$$

- Cuando S(i) tiende a 1 entonces $a(i)$<< $b(i)$ y, por tanto, el punto está muy bien clasificado.
- Cuando S(i) tiende a -1, entonces $a(i)$>> $b(i)$ y, por tanto, el punto estaría mejor en su *cluster* vecino.
- El promedio de S(i) sobre todos los puntos de un *cluster* nos informa de cómo de bien agrupados están.
- El promedio de los coeficientes de silueta de todos los puntos del conjunto de datos se puede usar para decidir el número óptimo de *clusters*:
    1. Por ejemplo, empezamos con 2 *clusters* y calculamos la silueta de cada cluster.
    2. Se calcula la silueta promedio de todo el conjunto de datos.
    3. Si la silueta promedio de algún *cluster* es muy inferior al promedio global, el número de *clusters* está por debajo de lo óptimo y debemos elegir uno más.
    4. Repetimos los pasos 1 a 3 para diferentes números de *clusters* hasta alcanzar un valor óptimo según el contexto o siluetas similares.

5. PROGRAMAS DE SOFTWARE LIBRE

Se presentan en este apartado los programas de software libre que se utilizarán posteriormente para realizar los ejemplos. La selección de estos se basa en el estudio realizado por Altalhi *et al.* (2017), eligiendo los que mayor puntuación alcanzan entre los que cuentan con una licencia GNU (Licencia Pública General).

Orange

https://orangedatamining.com/

- El software está desarrollado en la Universidad de Ljubljana en la Facultad de Informática. En este caso, está desarrollado en C++ y puede utilizarse mediante su entorno gráfico o conectándose con una *API* (Interfaz de Programación) desde Python.
- Su web cuenta con diversos recursos destinados al aprendizaje y consulta de dudas, como enlaces a video tutoriales propios en YouTube y un catálogo de *widgets* (operadores), con la descripción y ejemplos de uso de cada uno.
- Al iniciar el programa, nos encontramos con una ventana con accesos directos a esos recursos de aprendizaje:

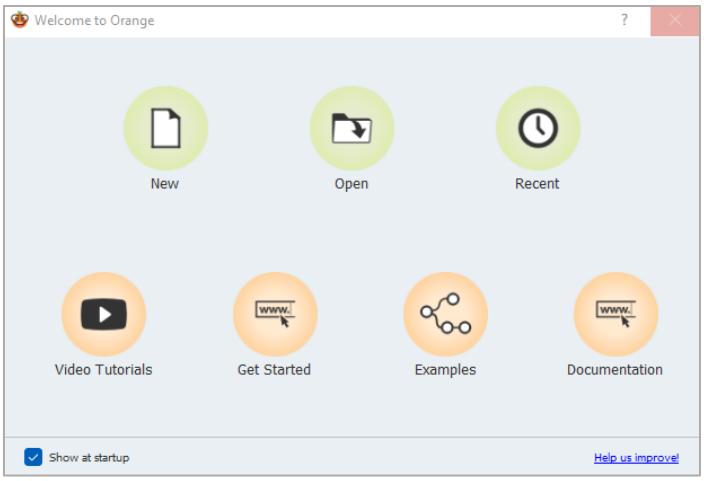


Figura 7. Ventana inicial de Orange.

- Su catálogo de operadores se muestra con una interfaz muy sencilla y agradable para el usuario:

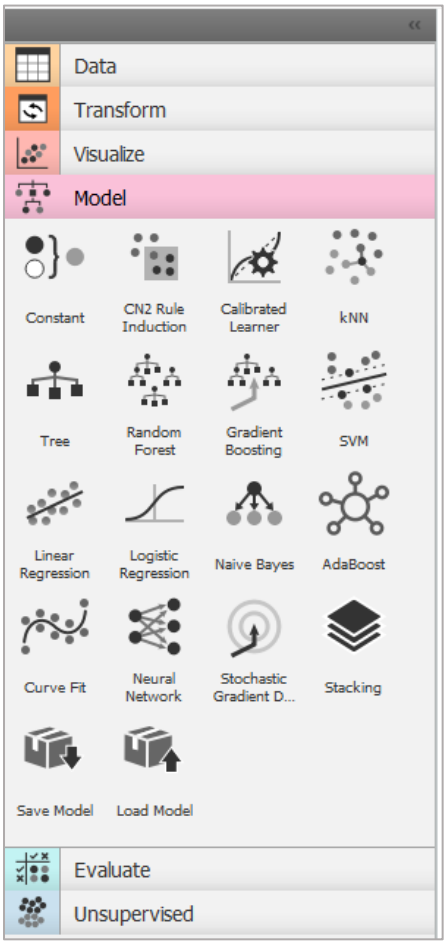


Figura 8. Componentes de Orange.

Knime

https://www.knime.com/

- Su nombre se corresponde con las siglas Konstanz Information Miner y se desarrolló en la Universidad de Constanza (Alemania), concretamente en el departamento de Bioinformática y Minería de Datos. Debido a su crecimiento, se ha terminado convirtiendo en una empresa situada en Zúrich, obteniendo ingresos con versiones de pago del programa, además de ofreciendo servicios de consultoría y formación para otras empresas.
- En su propia web, en un apartado llamado *hub* (https://hub.knime.com/), se pueden encontrar diversos tutoriales para comenzar a utilizar el programa, así como la descripción de todos los nodos y componentes disponibles a modo de ayuda.
- Como es habitual en este tipo de programas, su repositorio de nodos/operadores se encuentra en la parte inferior-izquierda en forma de un sencillo esquema para facilitar las búsquedas del usuario:

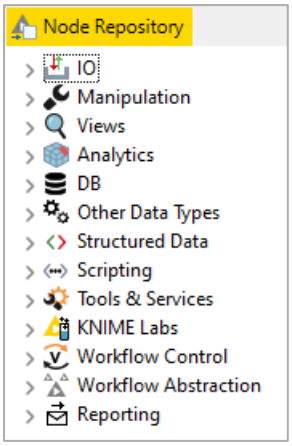


Figura 9. Componentes de Knime.

RapidMiner

https://rapidminer.com/

- Software creado en 2011 en el departamento de Inteligencia Artificial de la Universidad de Dortmund, desarrollado en Java. Puede utilizarse mediante su interfaz gráfica, ejecutando comandos con la consola que integra o con llamadas desde otros programas a través de sus *APIs*. En sus comienzos era conocido como YALE (Yet Another Learning Environment).
- En el momento de abrir el software por primera vez, se ofrece un tutorial para los primeros pasos en el programa, accesible en cualquier otro momento desde los menús:

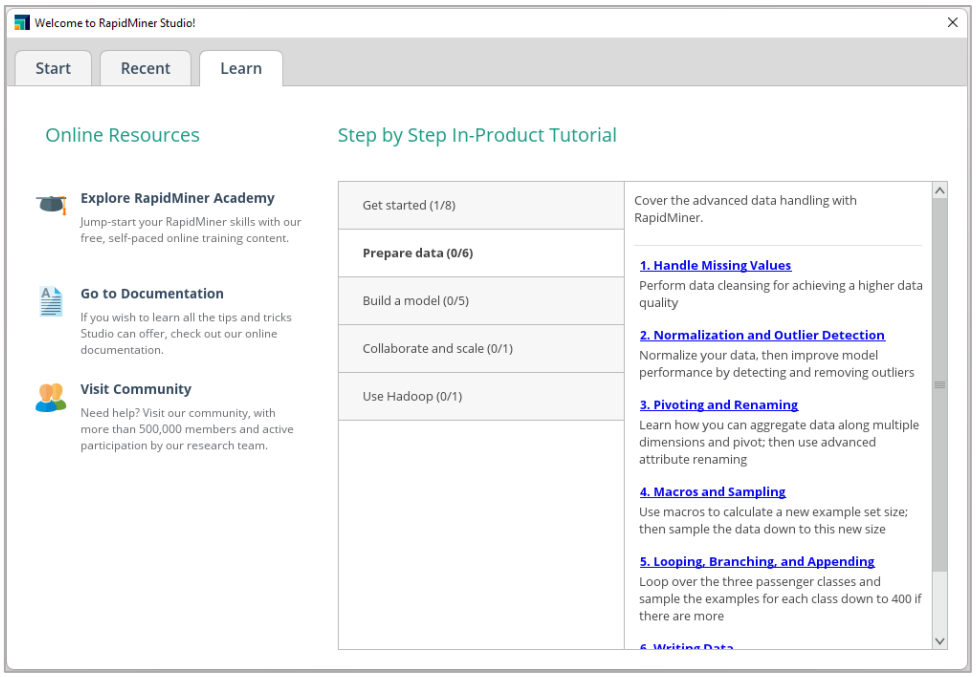


Figura 10. Ventana de inicio de RapidMiner

- Además, en su web se encuentra la documentación de cada uno de los objetos que se pueden utilizar (https://docs.rapidminer.com/latest/studio/getting-started/), con algunos ejemplos de aplicación que ayudan mucho a la hora de consultarla.
- Como en los casos anteriores, los operadores disponibles se encuentran en la esquina inferior izquierda:

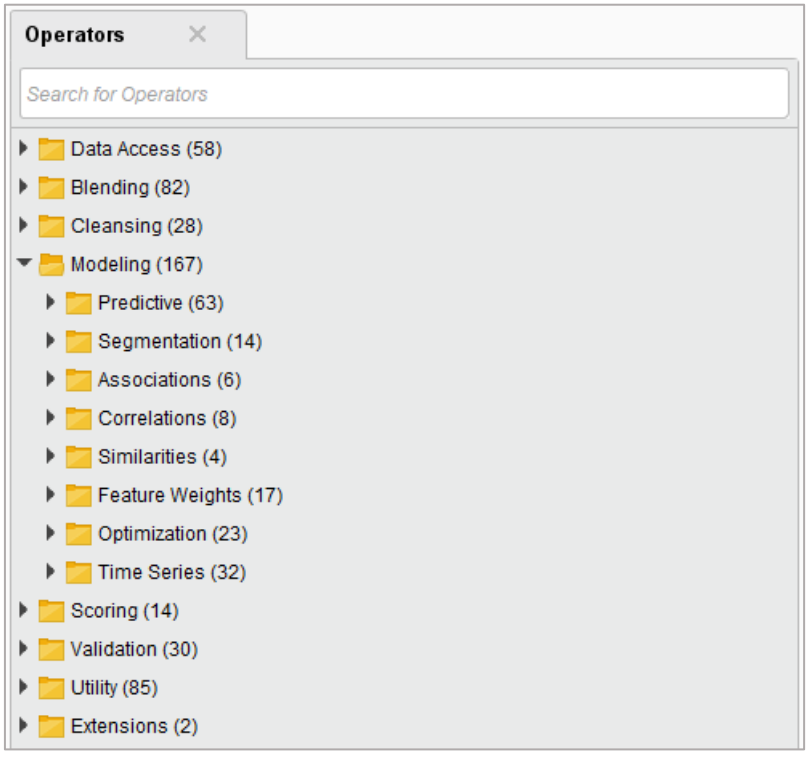


Figura 11. Componentes de RapidMiner

- Este software cuenta con una funcionalidad llamada *Auto Model* que consiste en un proceso guiado para construir un modelo según las necesidades, muy útil para iniciarse en su manejo y comprobar la potencia de las posibilidades que ofrece:

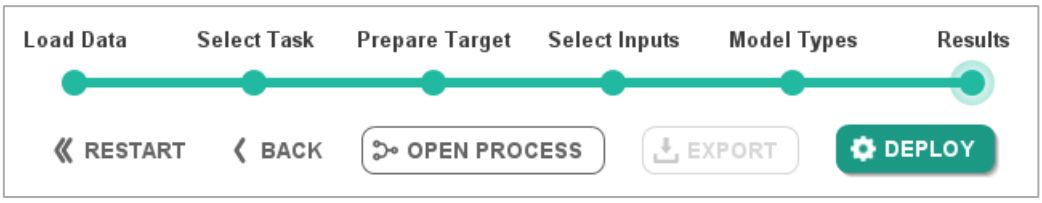


Figura 12. Pasos guiados de *Auto Model* de RapidMiner.

Weka

https://www.cs.waikato.ac.nz/ml/weka/

- Se trata de un programa de software libre creado por la Universidad de Waikato, de Nueva Zelanda, programada en Java que integra diversos algoritmos enfocados a la aplicación de la minería de datos y el *machine learning* (aprendizaje automático).
- Al abrir el programa se puede acceder a la sección de ayuda para dar los primeros pasos en su interfaz:

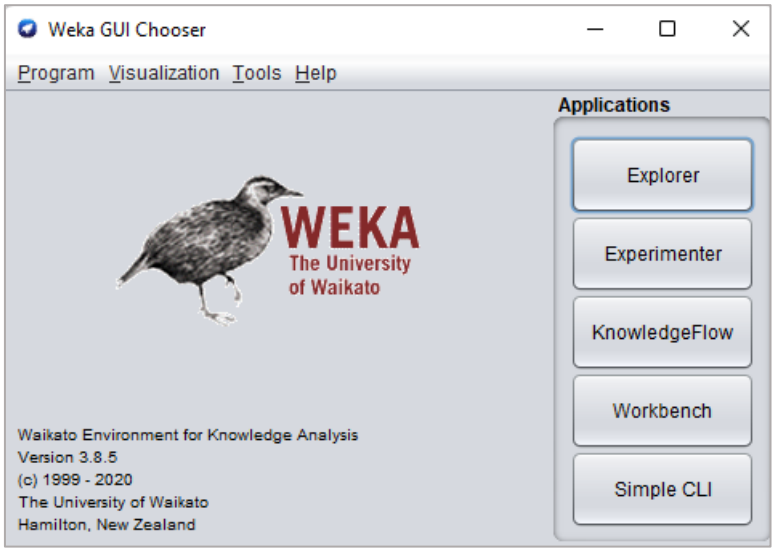


Figura 13. Ventana inicial de Weka.

- Al ser el programa más antiguo de los estudiados, su acceso a los diferentes componentes es algo distinto, mostrándose en forma de jerarquía de carpetas.

6. CASO DE ESTUDIO: CLASIFICACIÓN (SUPERVISADO)

Como ejemplo de aplicación se utiliza un conjunto de datos de reservas hoteleras en el que la variable objetivo será si la reserva se ha cancelado o no, tratándose así de un problema de clasificación binaria.

Los datos han sido obtenidos de la siguiente página web:

https://www.kaggle.com/datasets/datacertlaboratoria/proyecto-4-anlisis-de-cancelaciones-hoteleras

6.1. Descripción de los datos

Base de datos de cerca de 119.000 registros con datos históricos de reservas de una cadena hotelera. Cada fila representa una reserva. Los datos que se disponen son los siguientes:

Hotel - Tipo de hotel de la cadena. Puede ser resort u hotel de ciudad

is_canceled - La reserva fue cancelada (1) / La reserva no fue cancelada (0).

lead_time - Número de días transcurridos entre la reserva y la fecha de llegada.

arrival_date_year - Año de la fecha de llegada.

arrival_date_month - Mes de la fecha de llegada.

arrival_date_week_number - Número de semana del año para la fecha de llegada.

arrival_date_day_of_month - Día de la fecha de llegada.

stays_in_weekend_nights - Número de noches de fin de semana (sábado o domingo) que el huésped se alojó o reservó para quedarse en el hotel.

stays_in_week_nights - Número de noches de la semana (de lunes a viernes) que el huésped se hospedó o reservó para quedarse en el hotel.

adults - Número de adultos.

children - Número de niños.

babies - Número de bebés.

meal - Tipo de comida reservada.

country - País de origen.

market_segment - Designación del segmento de mercado. En las categorías, el término "TA" : Agentes de viajes y "TO" : Operadores turísticos.

distribution_channel - Canal de distribución de reservas. El término "TA" : Agentes de viajes y "TO" : Operadores turísticos.

is_repeated_guest - El nombre de la reserva era de un huésped repetido (1) / No era un huésped repetido (0)

previous_cancellations - Número de reservas anteriores que fueron canceladas por el cliente antes de la reserva actual

previous_bookings_not_canceled- Número de reservas anteriores no canceladas por el cliente antes de la reserva actual

reserved_room_type - Código del tipo de habitación reservada. El código se presenta en lugar de la designación por razones de anonimato

assigned_room_type - Código del tipo de habitación asignado a la reserva.

booking_changes - Número de cambios realizados en la reserva desde el momento en que se introdujo la reserva hasta la entrada

agent- Indicación de si el cliente hizo un depósito para garantizar la reserva. Esta variable puede asumir tres categorías: Sin depósito: no se realizó ningún depósito; No reembolsable: se realizó un depósito por el valor del coste total de la estancia; Reembolsable: se realizó un depósito por un valor inferior al coste total de la estancia.

company - El DNI de la agencia de viajes que hizo la reserva

days_in_waiting_list - ID de la empresa/entidad que realizó la reserva o responsable del pago de la misma. La identificación se presenta en lugar de la designación por razones de anonimato

customer_type - Número de días que la reserva estuvo en la lista de espera antes de ser confirmada al cliente

adr - Tipo de reserva, asumiendo una de las cuatro categorías: Contract - cuando la reserva tiene asociada una adjudicación u otro tipo de contrato; Group - cuando la reserva está asociada a un grupo; Transient - cuando la reserva no forma parte de un grupo o contrato, y no está asociada a otra reserva transitoria; Transient-party - cuando la reserva es transitoria, pero está asociada al menos a otra reserva transitoria

required_car_parking_spaces - La tarifa media diaria (adr por sus siglas en inglés) se define dividiendo la suma de todas las transacciones de alojamiento por el número total de noches de estancia

total_of_special_requests - Número de plazas de aparcamiento que necesita el cliente

reservation_status - Número de peticiones especiales realizadas por el cliente (por ejemplo, cama doble o piso alto)

reservation_status_date - Estado de la última reserva, asumiendo una de las tres categorías: Canceled - la reserva fue cancelada por el cliente; Check-Out -

el cliente se ha registrado, pero ya se ha marchado; No-Show - el cliente no se ha registrado y ha informado al hotel del motivo

6.2. Tratamiento con Orange

Proceso para construir el modelo:

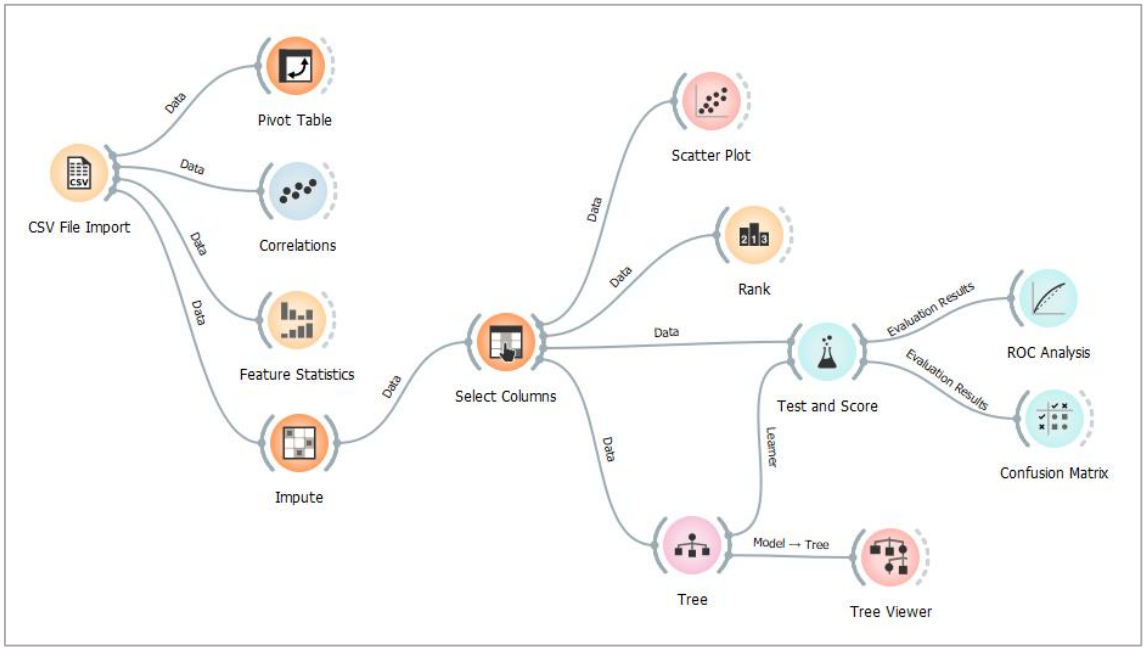


Figura 14. Proceso en Orange.

Con el nodo *Pivot Table* comprobamos la proporción de la clase a predecir para saber si estamos ante un claro desbalanceo entre clases o no:

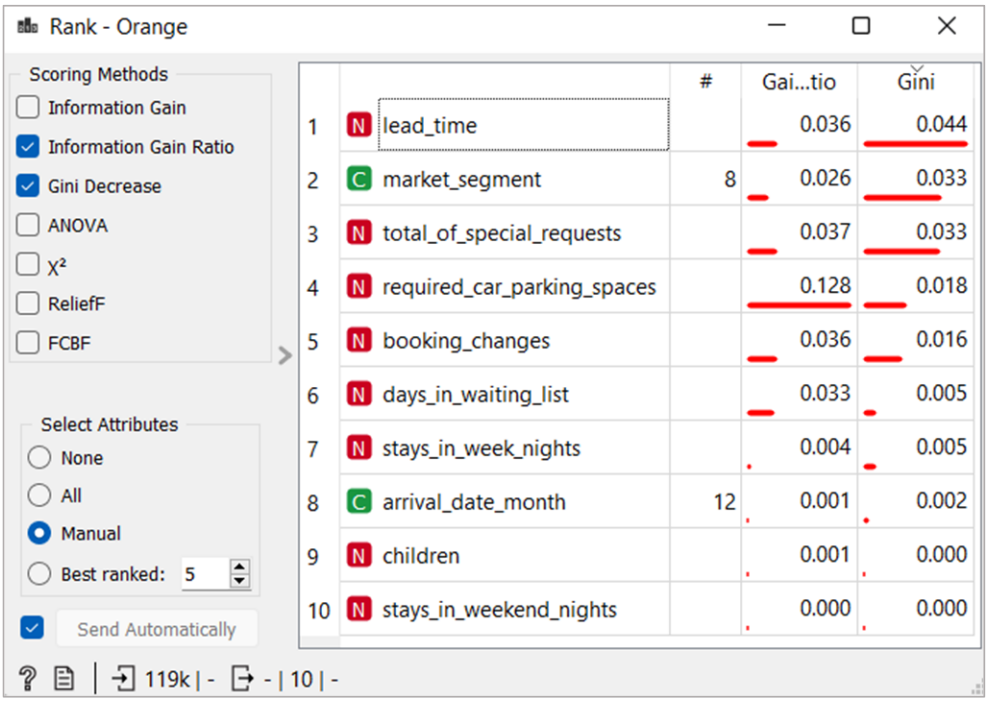
		is_canceled		
	Count	**0**	**1**	**Total**
reservation_status	**Canceled**	0.0	43017.0	**43017.0**
	Check-Out	75166.0	0.0	**75166.0**
	No-Show	0.0	1207.0	**1207.0**
	Total	**75166.0**	**44224.0**	**119390.0**

Figura 15. Reservas canceladas y no canceladas.

Al contar con un 37% de casos positivos, no lo vamos a considerar como un excesivo desbalanceo, por lo que podemos utilizar la *accuracy* como métrica principal de validación del modelo.

De esta tabla también se comprueba que el campo *reservation_status* aporta la misma información que la variable objetivo, motivo por el que no se debe incluir en el modelo. De todos modos, a continuación, se realiza una selección de variables.

Para simplificar el problema de clasificación y reducir las probabilidades de sobreajuste del modelo, con el nodo *Rank* se seleccionan las variables que pueden aportar más información para explicar la variable de salida. Se opta por elegir las 5 con mayor puntuación, que prácticamente coinciden al utilizar las métricas de *Information Gain Ratio* y *Gini Decrease:*



		#	Gai...tio	Gini
1	N lead_time		0.036	0.044
2	C market_segment	8	0.026	0.033
3	N total_of_special_requests		0.037	0.033
4	N required_car_parking_spaces		0.128	0.018
5	N booking_changes		0.036	0.016
6	N days_in_waiting_list		0.033	0.005
7	N stays_in_week_nights		0.004	0.005
8	C arrival_date_month	12	0.001	0.002
9	N children		0.001	0.000
10	N stays_in_weekend_nights		0.000	0.000

Figura 16. Tabla de salida del módulo *Rank* en Orange.

Las 5 variables que resultan ser las que más información aportan al modelo, son las siguientes (apoyándonos en el nodo *Feature Statistics* para su visualización):

1. El número de días transcurridos desde la reserva hasta la llegada (*lead_time*). Los clientes con menor propensión a cancelar la visita son los que la realizan con menos tiempo, como se aprecia en la distribución (en rojo las cancelaciones):

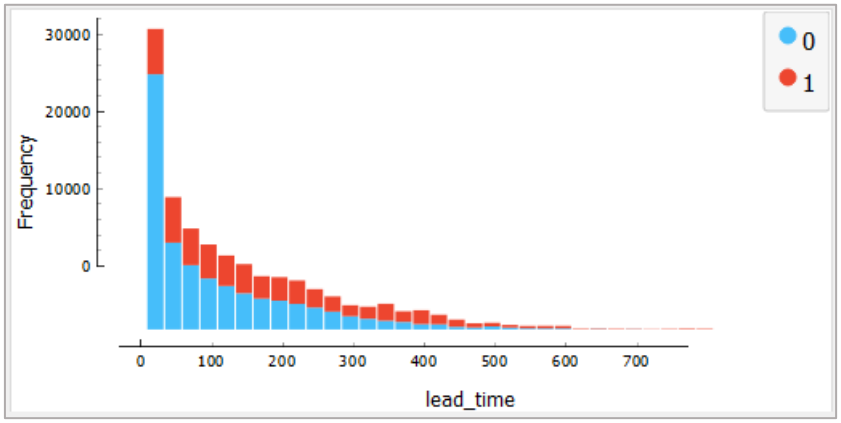


Figura 17. Distribución de *lead_time* segmentado por cancelación o no.

2. El segmento del mercado, es decir, el canal en el que el cliente realiza la reserva. Con *Pivot Table* se crea una tabla para mostrar que hay segmentos en los que es más propensa la cancelación, como Grupos, y otros en los que no, como en *Corporate*:

	is_canceled		
Count	**0**	**1**	**Total**
market_segment			
Aviation	185.0	52.0	**237.0**
Complementary	646.0	97.0	**743.0**
Corporate	4303.0	992.0	**5295.0**
Direct	10672.0	1934.0	**12606.0**
Groups	7714.0	12097.0	**19811.0**
Offline TA/TO	15908.0	8311.0	**24219.0**
Online TA	35738.0	20739.0	**56477.0**
Undefined	0.0	2.0	**2.0**
Total	**75166.0**	**44224.0**	**119390.0**

Figura 18. Tabla con las cancelaciones por segmento de mercado.

3. Total de peticiones especiales. A medida que aumentan las peticiones especiales del cliente, disminuye la propensión a la baja:

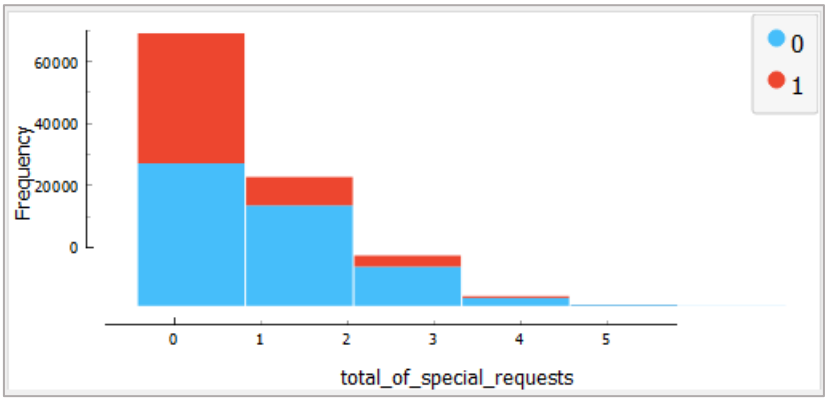


Figura 19. Distribución de *total_of_special_requests* segmentado por cancelación o no.

4. Plazas de parking solicitadas. Se comprueba que, aunque sean pocas instancias, al requerir una plaza de parking en la reserva la cancelación es 0:

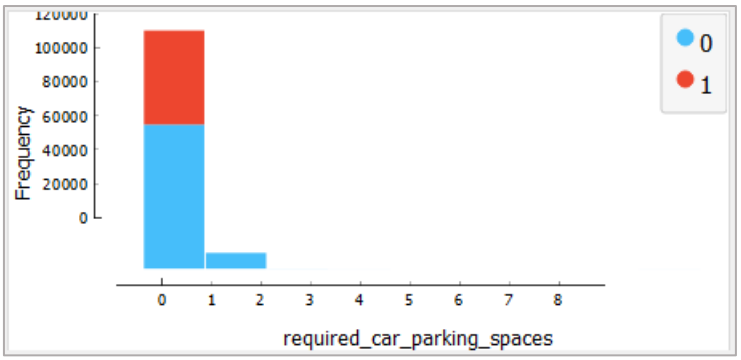


Figura 20. Distribución de plazas de parking segmentado por cancelación o no.

5. Cambios realizados en la reserva. A medida que aumentan los cambios solicitados en la reserva, disminuye la propensión a la cancelación. Se interpreta que cuando se realizan cambios, el cliente tiene verdadero interés en acudir al hotel:

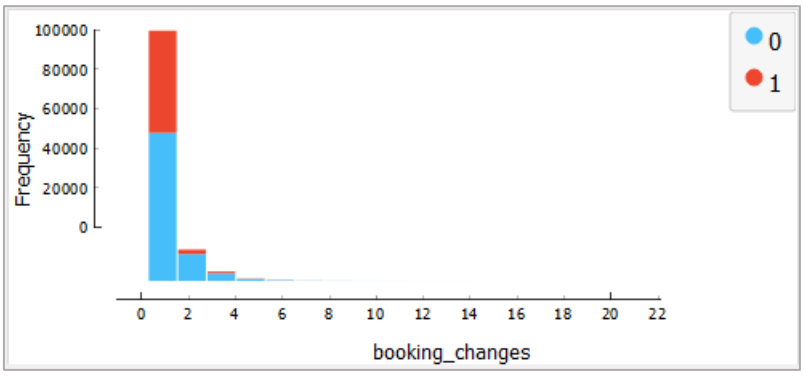


Figura 21. Distribución de cambios en la reserva segmentado por cancelación o no.

Este análisis de variables más significativas no se vuelve a realizar en el resto de los programas, utilizando los resultados aquí obtenidos como referencia para replicar la selección en el resto.

Se seleccionan esas 5 variables en el nodo *Select Columns* para entrenar el modelo.

En el nodo *Tree* se genera el árbol con todo el conjunto de datos con los parámetros por defecto, reduciendo el valor *maximal tree depth* de 100 a 50:

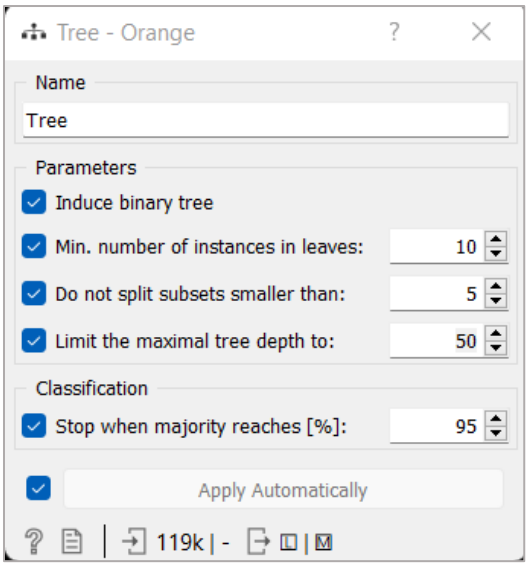


Figura 22. Opciones de configuración del nodo *Tree* en Orange.

La forma de evaluar el modelo en el nodo *Test and Score* por defecto es con una validación cruzada en 5 particiones "estratificadas", es decir, con la clase a predecir balanceada en cada una de las particiones:

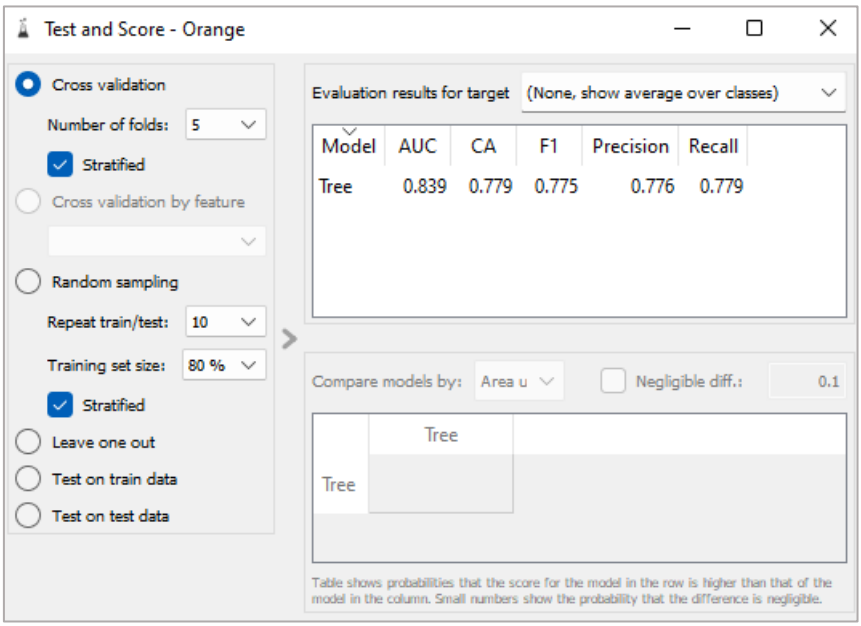


Figura 23. Salida del nodo *Test and Score* en Orange.

Resultado de la matriz de confusión tanto en porcentajes como en valores absolutos:

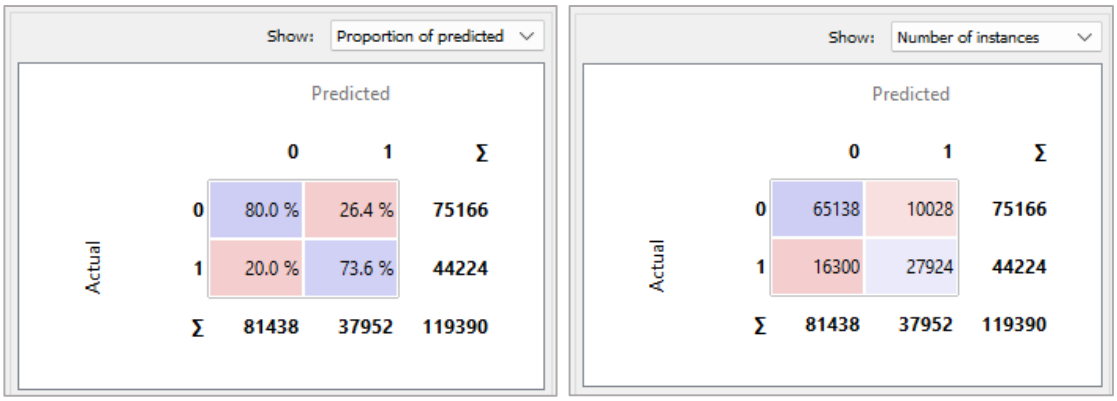


Figura 24. Matrices de confusión en Orange con los resultados de la clasificación.

Alcanzando una exactitud del 77,94%, al sumar el número de instancias clasificadas correctamente (65.138 + 27.924) entre las instancias totales (119.390).

Antes de realizar la selección de variables, se implementa un modelo con todas las variables y se logra una *accuracy* del 80%. La diferencia de menos del 3% utilizando sólo 5 variables es asumible al implementar un árbol menos complejo, consiguiendo un modelo que evite el sobreajuste.

Al estar implementado directamente en un componente del programa, se representa la curva ROC con el nodo *ROC Analysis*, donde se visualiza el buen resultado obtenido al acercarse la curva a la esquina superior izquierda:

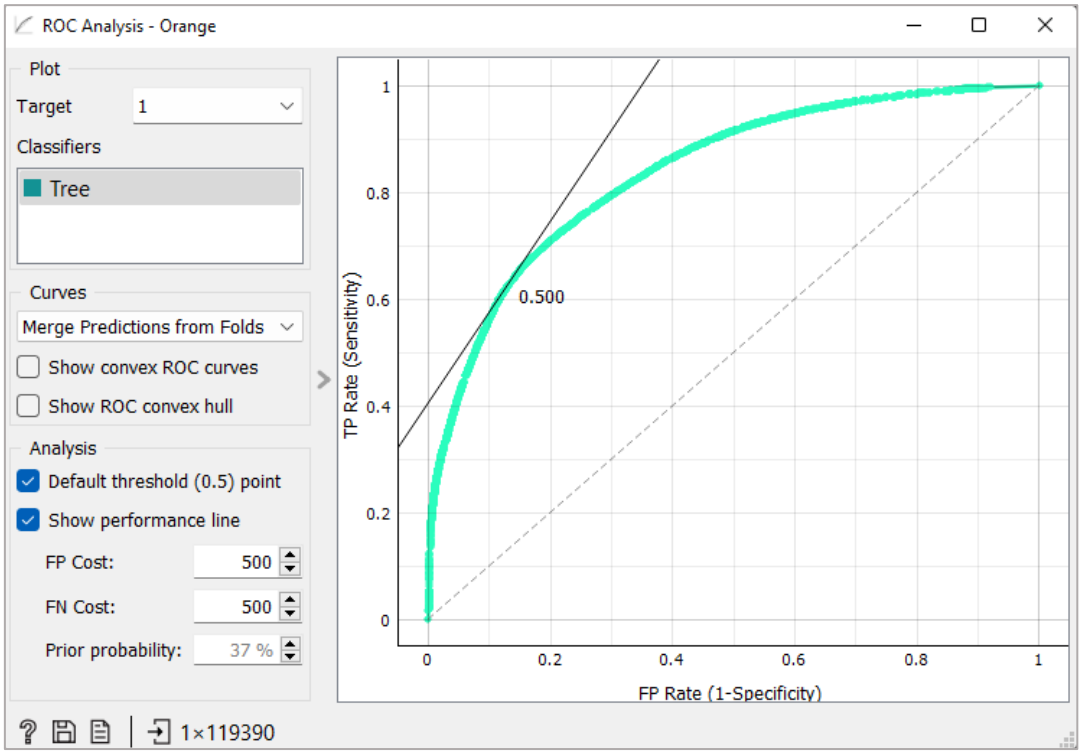


Figura 25. Curva ROC representada con Orange.

En el nodo *Tree Viewer* se representa el árbol de decisión del modelo, solamente los 4 primeros niveles debido a la complejidad del mismo, pero suficiente para comprobar que se considera la variable *lead_time* como la más importante del modelo al realizar la primera división del árbol:

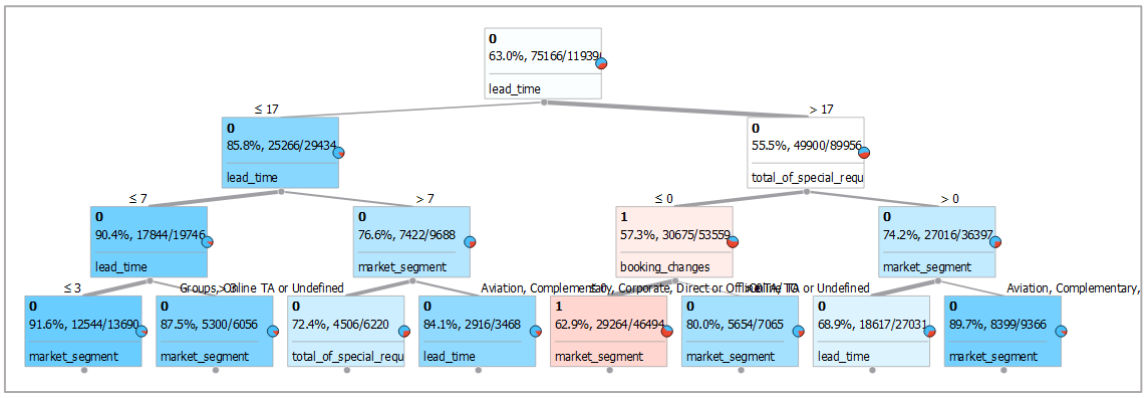


Figura 26. Representación parcial del árbol de decisión en Orange.

6.3. Tratamiento con Knime

Proceso de configuración en Knime:

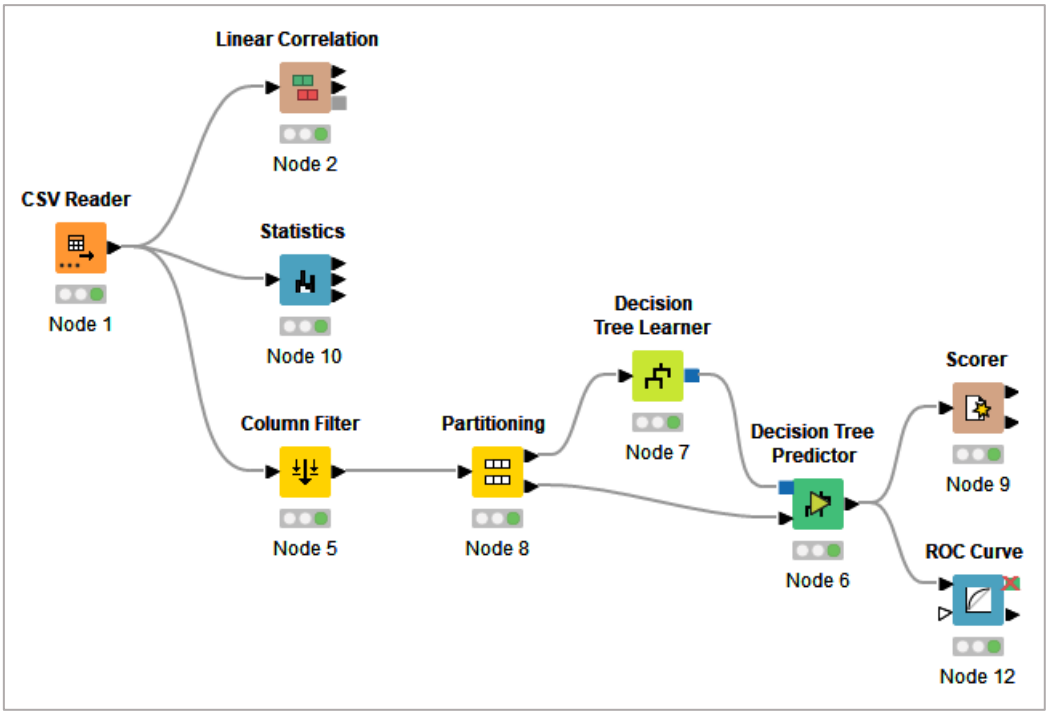


Figura 27. Proceso seguido en Knime.

- Con *Column Filter* se seleccionan las mismas variables que en el proceso implementado en Orange:

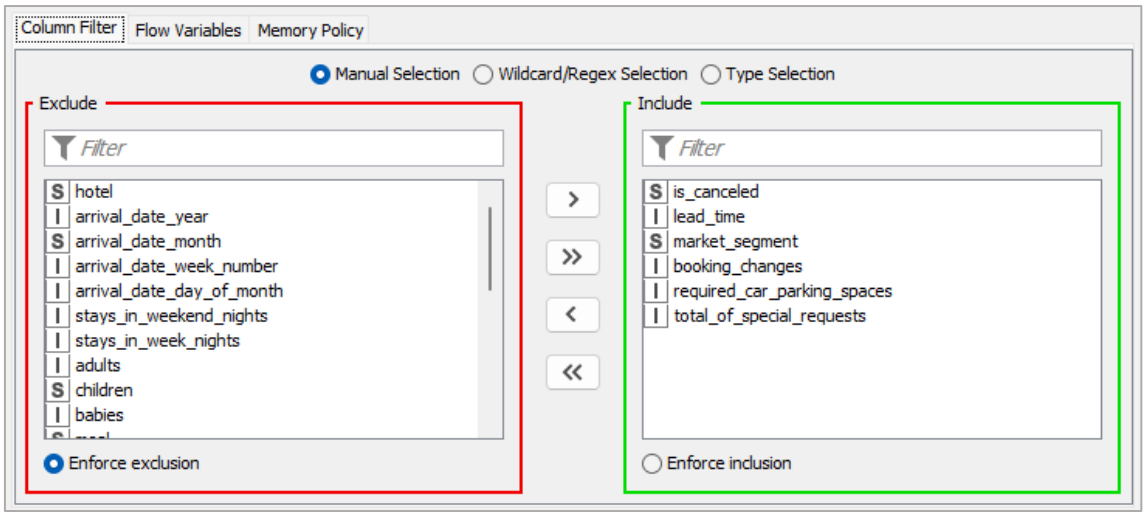


Figura 28. Selección de variables en Knime.

- Con el nodo *Partitioning* se divide el conjunto de datos en entrenamiento (*train*) y test con una proporción del 80% y el 20%, respectivamente.
- En el componente *Decision Tree Learner* se configuran los hiperparámetros del árbol a entrenar con el 80% del conjunto de datos, dejando los valores por defecto y ajustando a 10 instancias mínimas por nodo como en Orange:

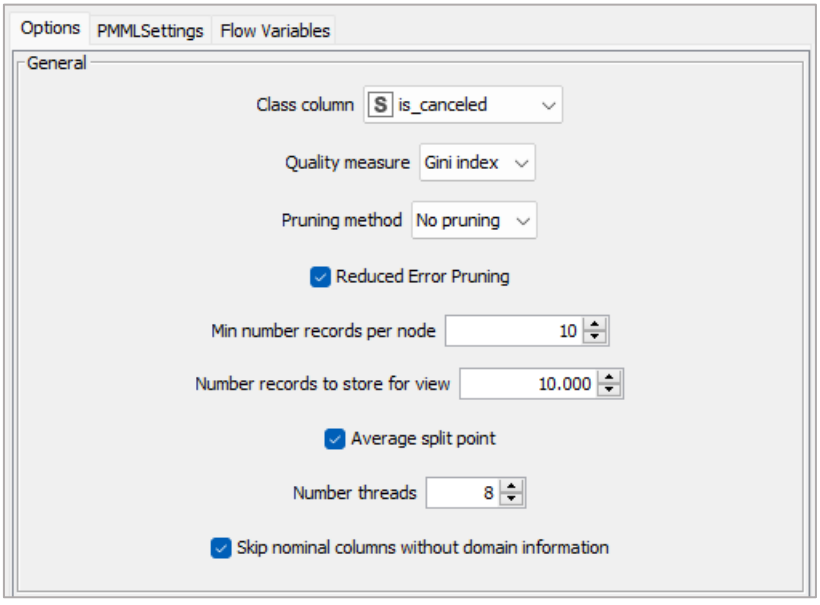


Figura 29. Opciones de configuración del árbol en Knime.

Los nodos iniciales del árbol se pueden visualizar en el componente *Decision Tree Predictor*, que tiene como entradas el árbol aprendido por el modelo anterior y el 20% restante del conjunto de datos:

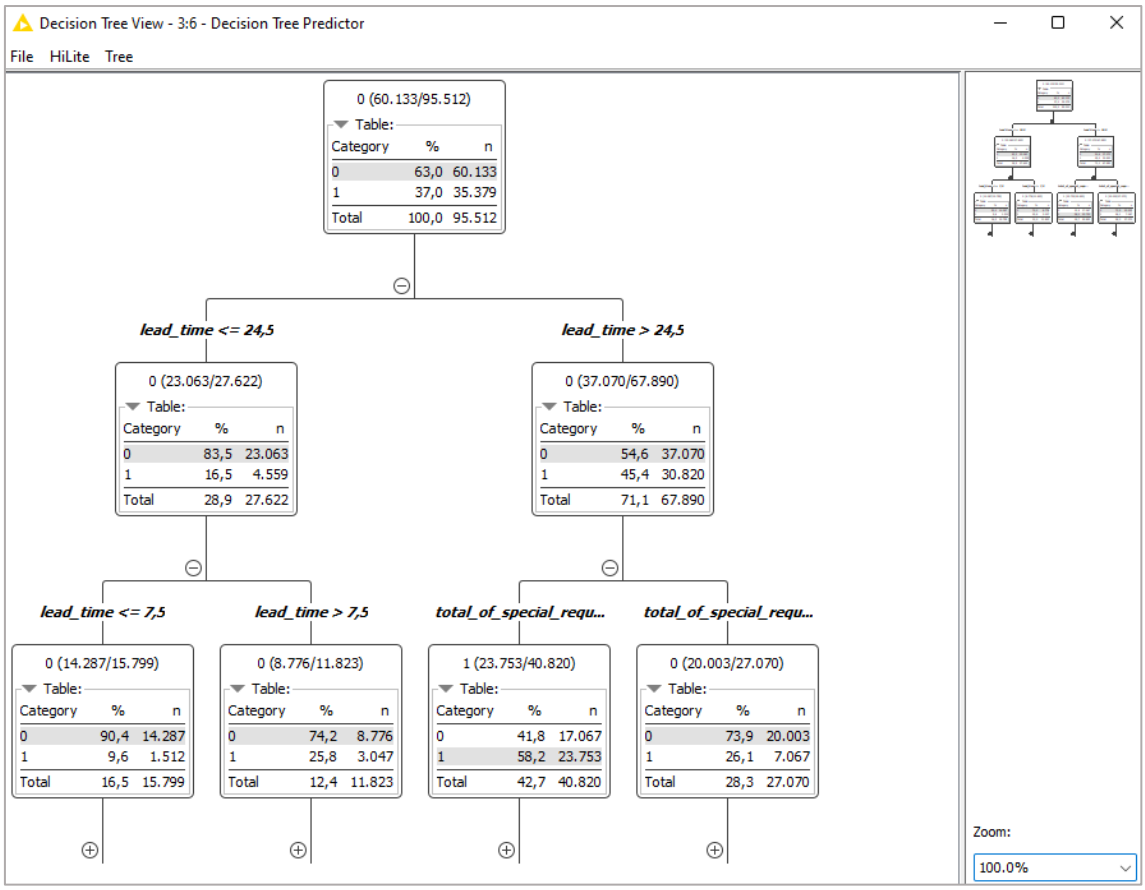


Figura 30. Representación parcial del árbol de decisión en Knime.

Volviendo a ser la variable *lead_time* la que realiza la primera división en los datos.

El *Scorer* sobre el conjunto de *test* resulta la siguiente matriz de confusión:

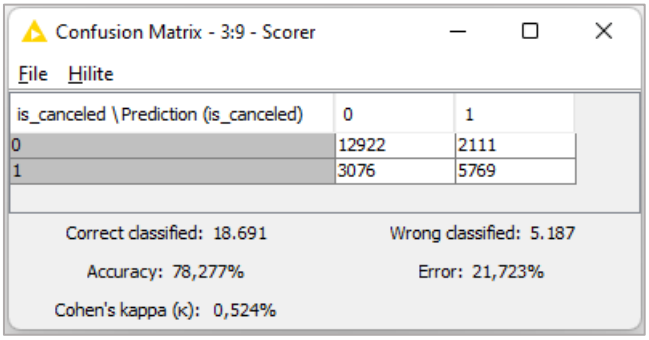


Figura 31. Matriz de confusión y otras puntuaciones conseguidas en Knime.

Clasificando correctamente un 78,27% de las observaciones, muy similar al obtenido en Orange.

Por último, se puede mostrar la curva ROC en el nodo *ROC Curve*, además del área AUC que es de 0.843 en este caso, curva que de nuevo se acerca a la esquina superior izquierda, consecuencia de haber construido un buen modelo de clasificación:

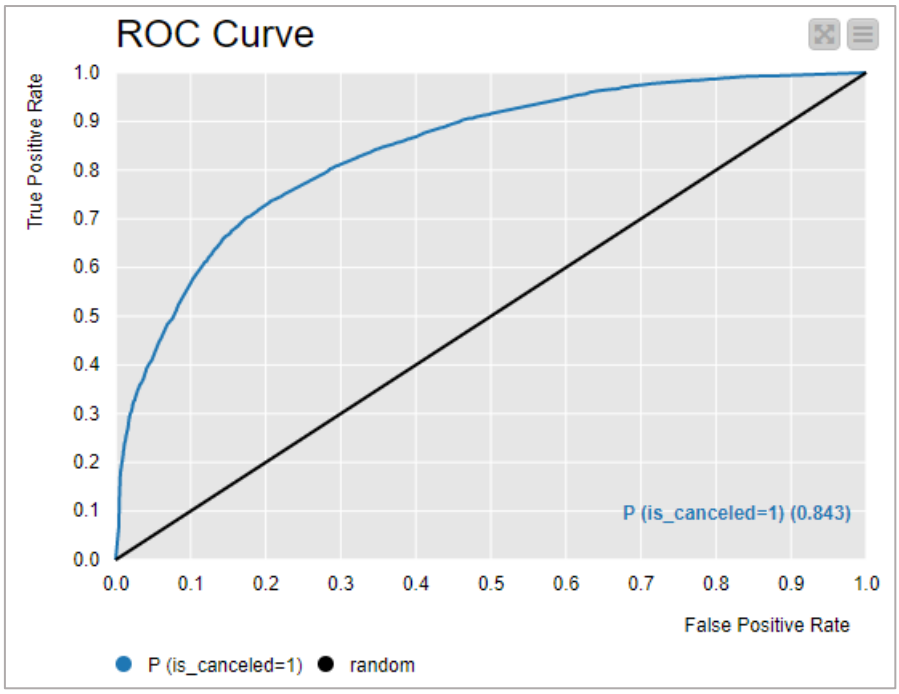


Figura 32. Representación de la curva ROC en Knime.

6.4. Tratamiento con RapidMiner

Proceso construido y breve explicación de alguno de los pasos:

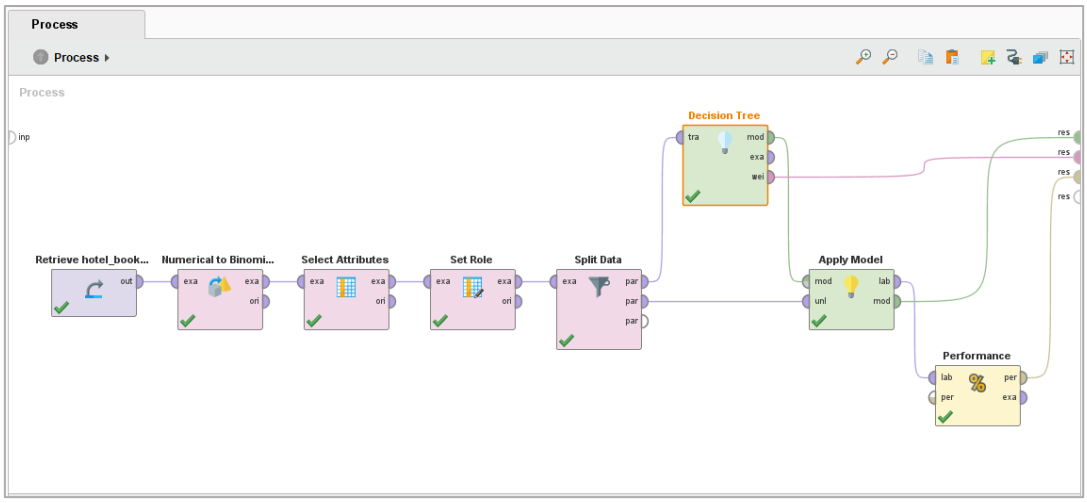


Figura 33. Proceso seguido en RapidMiner para crear el modelo de clasificación

- Se seleccionan los mismos atributos que con Orange en el nodo *Select Attributes*.
- En *Numerical to Binomial* es necesario indicarle al modelo que la variable objetivo es binaria.
- En *Set Role* se indica que la variable *is_canceled* es de tipo *label* al ser el objetivo de la predicción.

- Se divide el conjunto de datos en *train* y *test* en el nodo *Split Data*, en un 80% y 20%, respectivamente.

Parámetros por defecto del árbol, con 10 niveles de profundidad máximos:

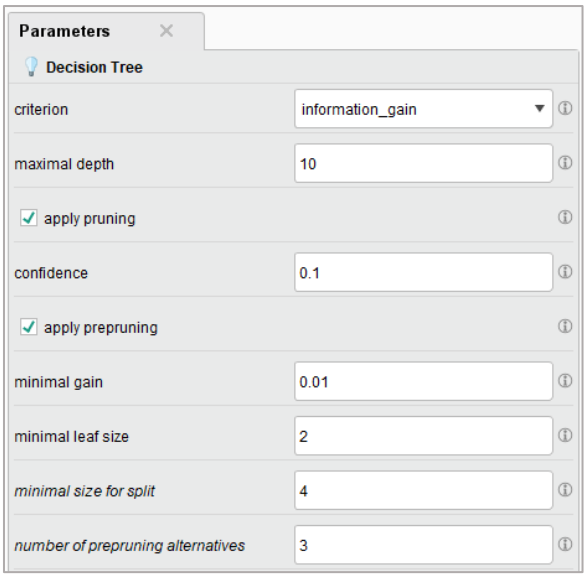


Figura 34. Opciones de configuración del árbol en RapidMiner.

La salida del nodo *Apply Model* muestra la representación del árbol, pudiendo elegir entre la versión gráfica, siendo más compleja que en los anteriores programas al no poder comprimir los nodos, y una versión esquemática como se muestra en la siguiente figura:

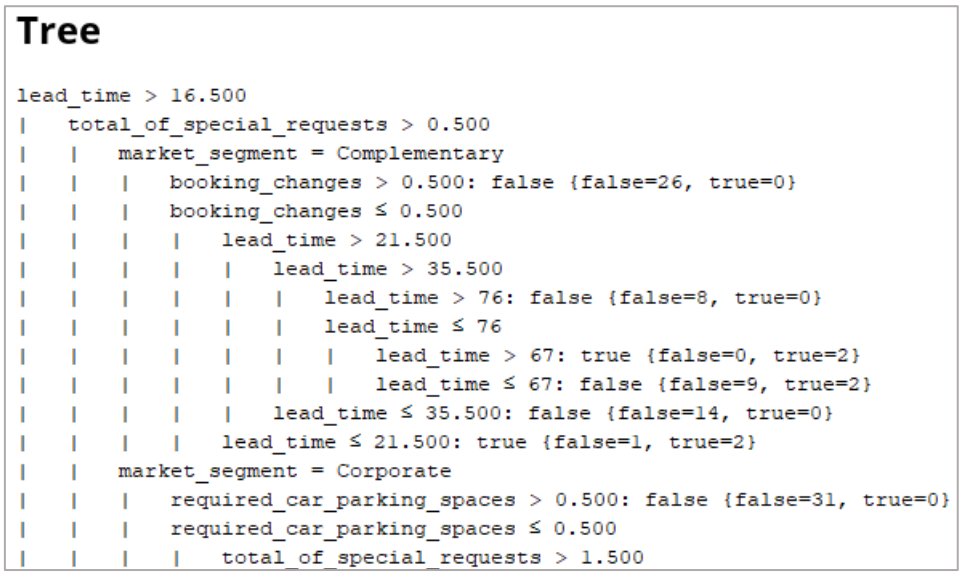


Figura 35. Representación esquemática del árbol de decisión en RapidMiner.

Suficiente para constatar que, de nuevo, al implementar el árbol de decisión en este software, la primera división de los datos se lleva a cabo mediante un nodo con la variable *lead_time*.

Es interesante comprobar que en otra de las salidas del modelo se puede presentar los pesos que el árbol otorga a cada variable para confirmar que *lead_time* es el atributo más importante para el árbol de decisión:

attribute	weight
required_car_parking_spaces	0.053
booking_changes	0.152
market_segment	0.029
lead_time	0.746
total_of_special_requests	0.020

Figura 36. Tabla con la importancia de las variables en RapidMiner.

Se puede observar la matriz de confusión con la salida del nodo *Performance,* obteniéndose una exactitud del 76.71% al probar el modelo con el conjunto de *test*:

accuracy: 76.71%

	true false	true true	class precision
pred. false	12732	3260	79.61%
pred. true	2301	5585	70.82%
class recall	84.69%	63.14%	

Figura 37. Matriz de confusión en RapidMiner.

Por último, es posible representar la curva ROC y el valor del área bajo la curva (AUC), donde de nuevo se aprecia el buen resultado del árbol al acercarse la curva a la esquina superior izquierda:

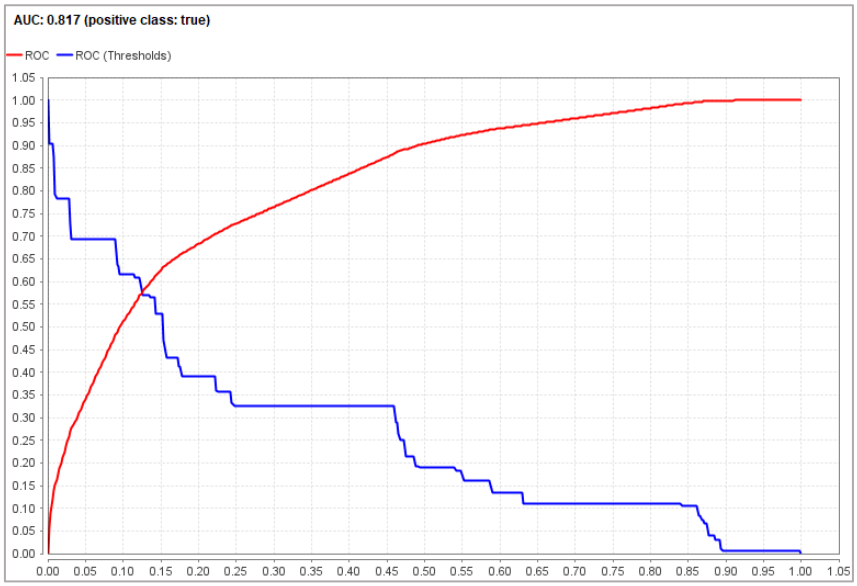


Figura 38. Curva ROC representada en RapidMiner.

6.5. Tratamiento con Weka

Con este software, en su aplicación *Explorer*, se procesan los datos encadenando opciones en los botones y menús, en lugar de seguir un proceso uniendo nodos como en los otros 3 programas.

El primer paso consistiría en filtrar los datos por las variables seleccionadas en Orange para replicar el modelo para que los resultados sean homogéneos y transformar la variable objetivo (*is_canceled*) en nominal:

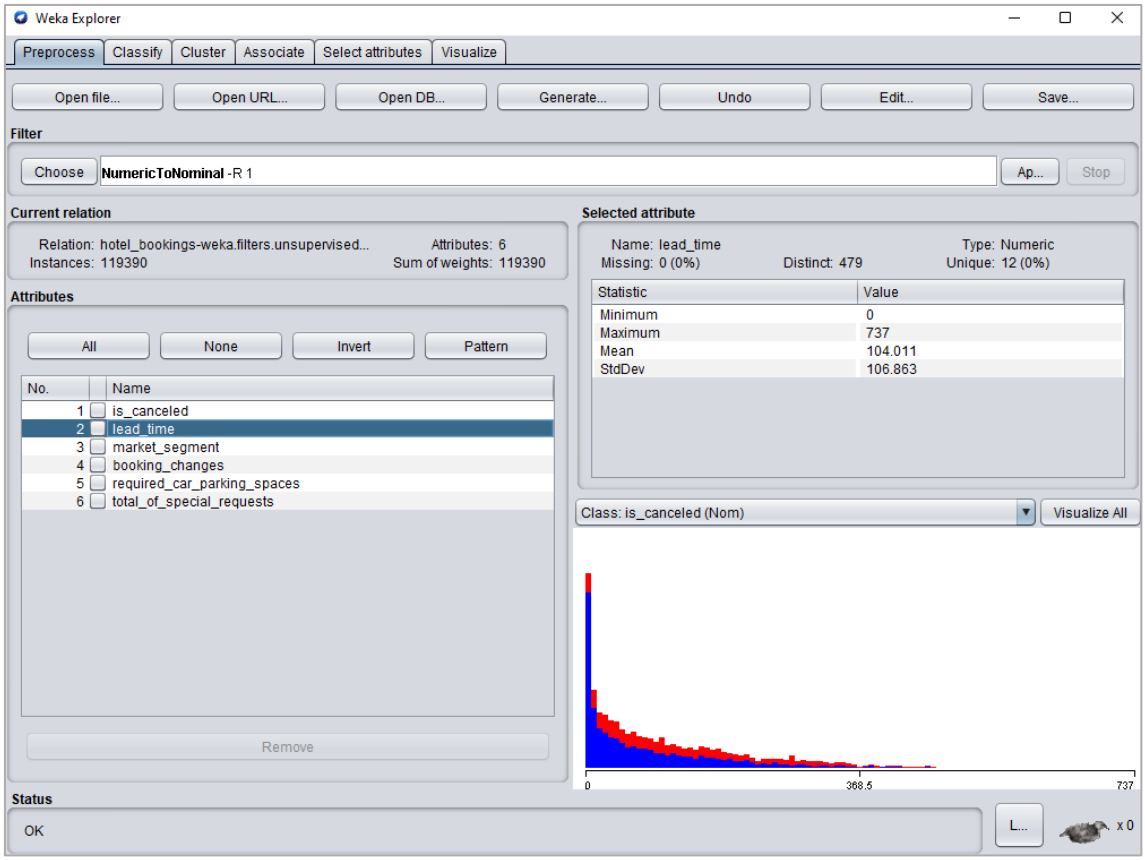


Figura 39. Ventana principal de Weka tras el filtrado de variables.

Eligiendo en la pestaña *Classify* el árbol a implementar (J48) con los valores por defecto:

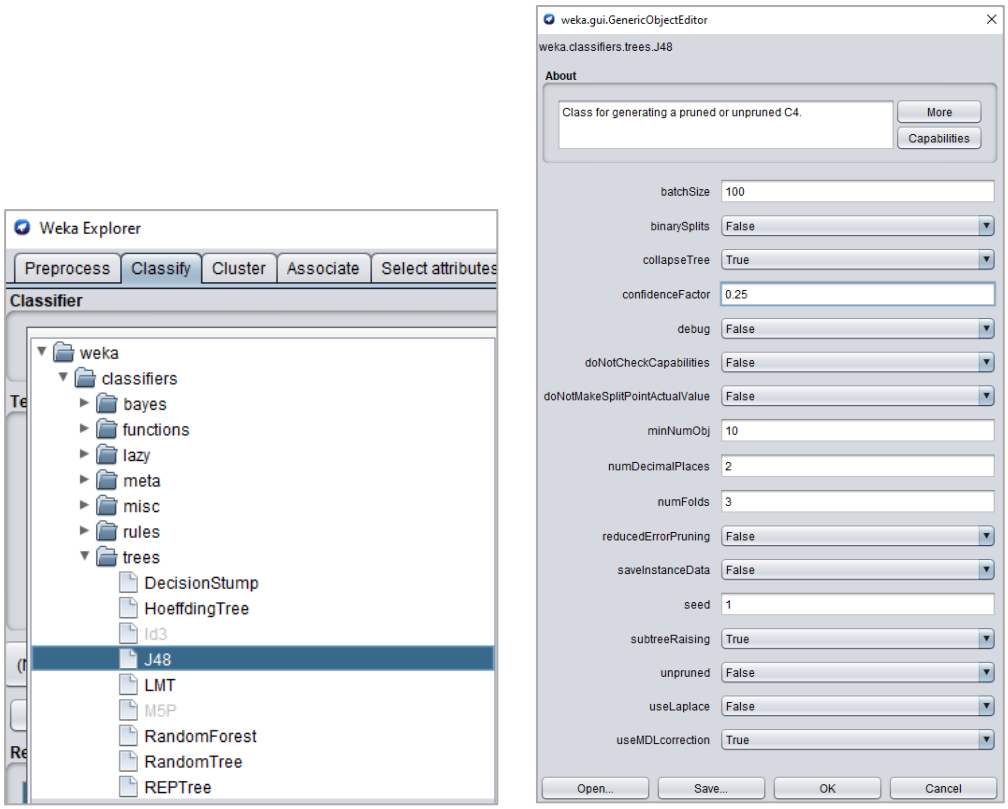


Figura 40. Componentes disponibles en Weka y opciones de configuración del árbol.

Seleccionando como objetivo la variable *is_canceled* con una validación cruzada con 10 *folds*:

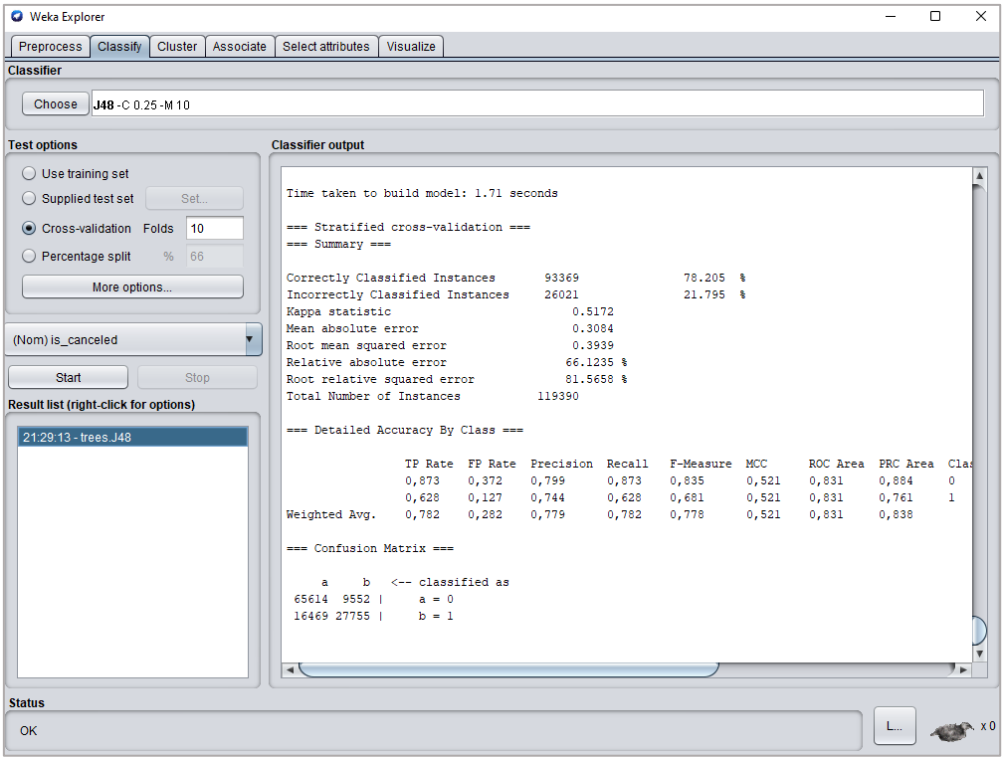


Figura 41. Ventana con los resultados obtenidos por el modelo en Weka (I).

La *accuracy* obtenida es del 78.20%, pudiendo observar en el esquema del árbol que, de nuevo, la variable que divide los datos por primera vez es *lead_time*:

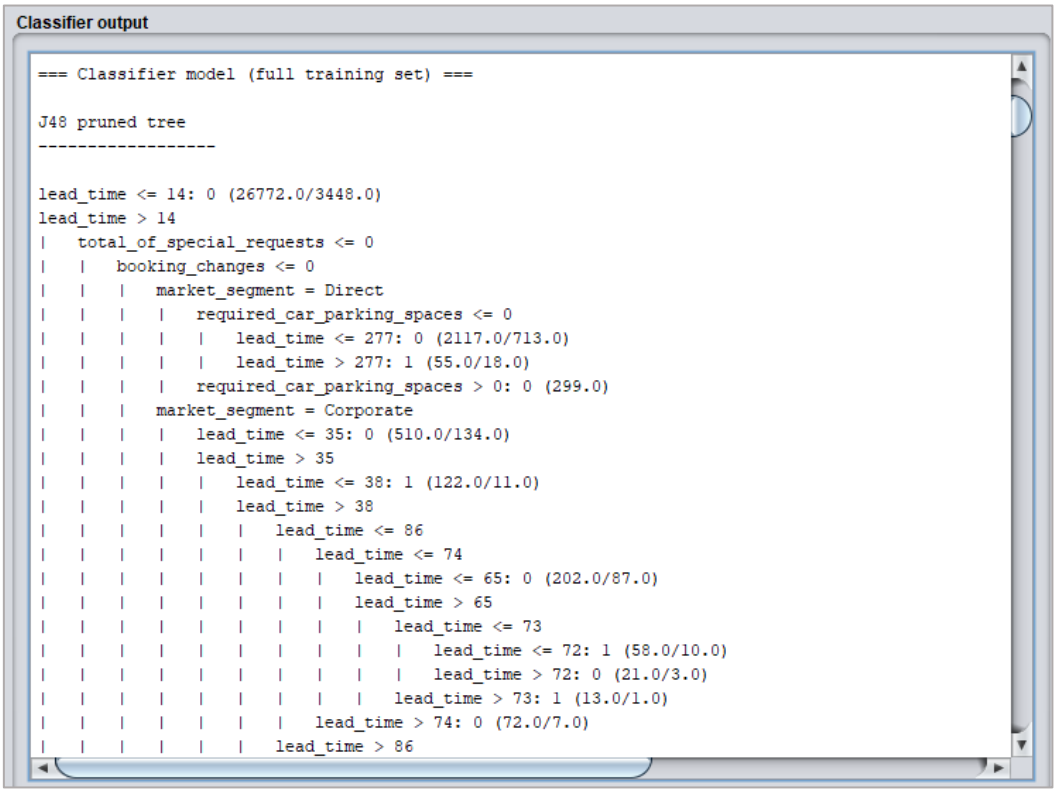


Figura 42. Ventana con los resultados obtenidos por el modelo en Weka (II).

En resumen, tras implementar un modelo clasificador en cada software, se pueden destacar los siguientes puntos en común:

- Resultados muy similares en las métricas utilizadas (comparativa realizada en el apartado 8.1.1) y en las curvas ROC.
- Se sigue logrando un buen resultado en las métricas tras reducir el número de variables independientes de 30 a solamente 5, reduciendo la complejidad del árbol.
- Pese a esa disminución en la complejidad, y para que lograse obtener unas métricas aceptables, el árbol continúa siendo demasiado profundo para poder representarse en este documento. No obstante, se observan las siguientes características comunes a todos los modelos:
    - El primer nodo que divide el conjunto de datos siempre es con el atributo *lead_time* (días desde que se realiza la reserva hasta la llegada al hotel), siendo ésta la variable con mayor importancia.
    - Las reservas que quedan clasificadas por encima del número de días definido en el primer nodo se vuelven a dividir según si tienen o no peticiones especiales, siendo ésta otra variable importante en los distintos árboles. Su interpretación a nivel de negocio es que, aunque el cliente reserve con mucha antelación y

esto suponga una mayor propensión a la cancelación, tener peticiones especiales reduce esa probabilidad de cancelar la reserva.

- En los siguientes niveles del árbol ya aparecen las variables de *market_segment*, *booking_changes* y, en niveles inferiores al aparecer en menos reservas, si requiere plazas de parking.

7. CASO DE ESTUDIO: CLUSTERING (NO SUPERVISADO)

Para ilustrar de forma práctica un problema de *clustering* se utilizará un conjunto de datos que contine diferentes características de 3 vinos distintos.

Fuente de datos: Frank A, Asuncion A (2010). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Wine

Descarga: https://www.kaggle.com/datasets/harrywang/wine-dataset-for-clustering/download

7.1. Descripción de los datos

Estos datos son los resultados de un análisis químico de vinos cultivados en la misma región de Italia, pero derivados de tres cultivos diferentes. El análisis determinó las cantidades de 13 componentes que se encuentran en cada uno de los tres tipos de vinos.

Alcohol – Cantidad de alcohol.

Malic acid - Es un tipo de ácido con fuerte acidez y aroma a manzana. El vino tinto es naturalmente acompañado de ácido málico.

Ash – Tiene un efecto sobre el sabor general del vino y puede darle una sensación fresca.

Alcalinity of ash - Es una medida de alcalinidad débil disuelta en agua.

Magnesium - Es un elemento esencial del cuerpo humano, que puede promover el metabolismo energético y es débilmente alcalino.

Total phenols - Moléculas que contienen sustancias polifenólicas, que tienen un sabor amargo y afectan el sabor, el color y sabor del vino, y pertenecen a los nutrientes del vino.

Flavanoids - Es un componente antioxidante para el corazón y antienvejecimiento, rico en aroma y amargo.

Nonflavanoid phenols - Es un gas aromático especial con resistencia a la oxidación y es débilmente ácido.

Proanthocyanins - Es un compuesto bioflavonoide, que también es un antioxidante natural con un ligero olor amargo.

Color intensity - Grado de tonalidad del color. Se utiliza para medir el estilo del vino para ser "ligero" o "grueso". La intensidad del color es alta, mientras más tiempo el vino y el mosto están en contacto durante el proceso de elaboración del vino, más denso es el sabor.

Hue - Se refiere a la viveza del color (matiz) y el grado de calor y frialdad. Se puede utilizar para medir la variedad y la edad del vino.

OD280/OD315 of diluted wines - Es un método para determinar la concentración de proteínas, que puede determinar el contenido proteico de varios vinos.

Proline - Es el aminoácido principal en el vino tinto y una parte importante de la nutrición y el sabor del vino.

Según la documentación, los 3 tipos de vino se reparten de la siguiente forma:

- Clase 1 – 59 instancias
- Clase 2 – 71 instancias
- Clase 3 – 48 instancias

Por lo que el objetivo será crear una división de 3 *clusters* y comparar con las proporciones de la división original.

7.2. Tratamiento con Orange

Proceso construido en el software:

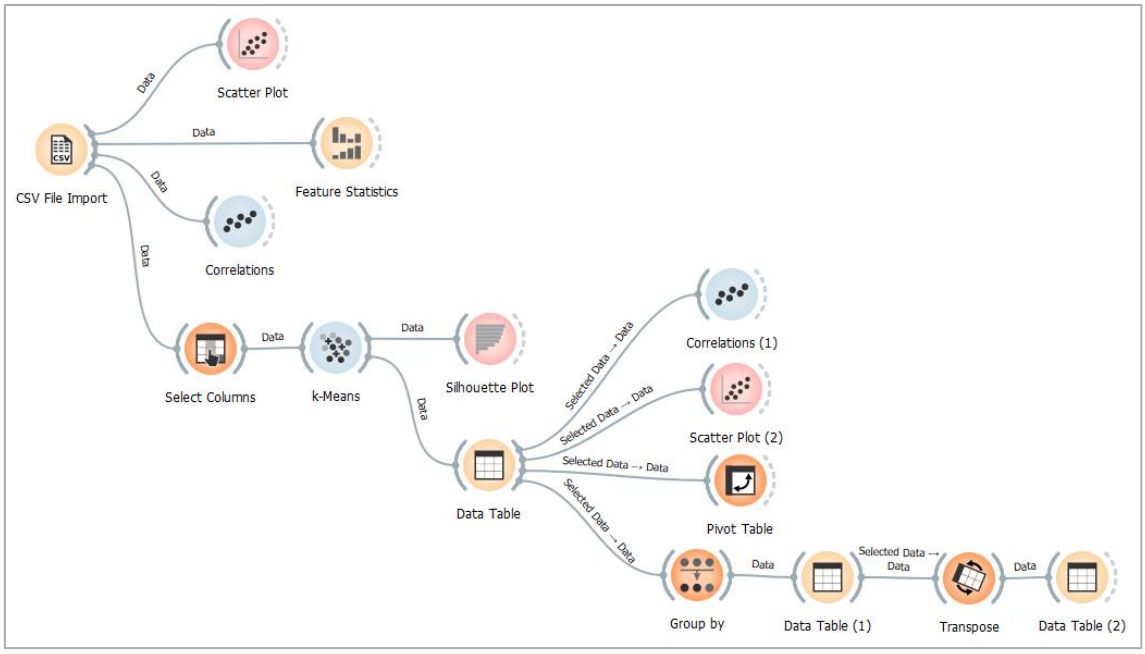


Figura 43. Proceso construido en Orange.

De forma sencilla se puede observar que el número óptimo de clústeres coincide con 3 gracias a la métrica *Silhouette Scores* facilitada por defecto en el componente *k-Means*, al realizarlo sobre un intervalo entre 2 y 8:

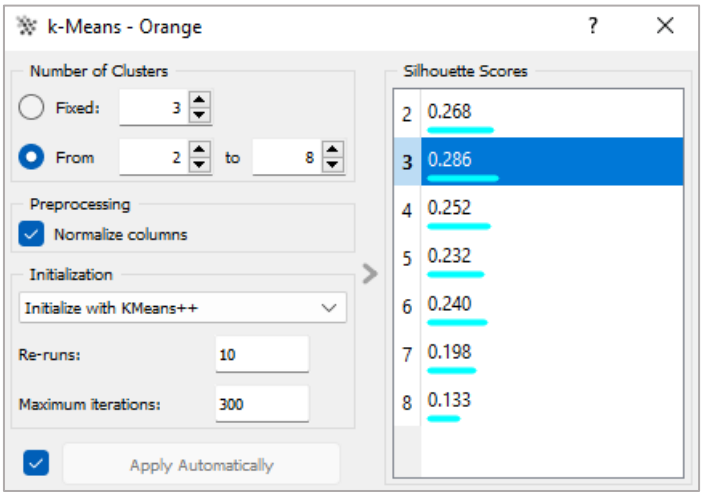


Figura 44. Resultado del nodo *k-Means* en Orange que incluye la métrica de silueta.

En el nodo *Pivot Table* se comprueba la cantidad de instancias por *cluster*, muy similar a la distribución original:

Cluster

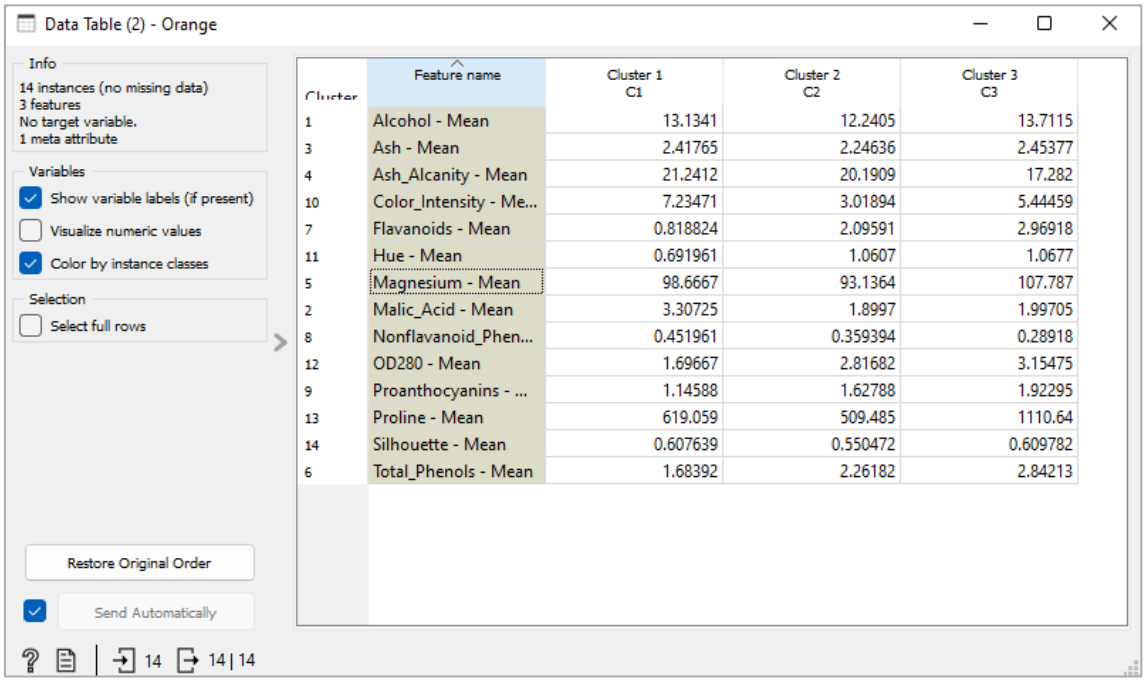
Count	C1	C2	C3	Total
C1	51.0	0.0	0.0	**51.0**
C2	0.0	66.0	0.0	**66.0**
C3	0.0	0.0	61.0	**61.0**
Total	**51.0**	**66.0**	**61.0**	**178.0**

Figura 45. Cantidad de instancias por *cluster* en Orange.

En *Scatter Plot (2)* se puede representar una nube de puntos asignando *Alcohol* al eje *X* y *Flavanoids* al eje *Y* para comprobar que con esas 2 variables (curiosamente un componente nocivo y otro beneficioso para el organismo) los *clusters* quedan definidos casi en su totalidad:

Figura 46. Nube de puntos representando 2 variables y los *clusters* en Orange.

El proceso que sigue a partir del nodo *Group by* tiene como finalidad poder presentar en una tabla las características de cada *cluster* según las medias de cada atributo de las instancias que han sido asignadas a cada uno de los 3:



Cluster	Feature name	Cluster 1 C1	Cluster 2 C2	Cluster 3 C3
1	Alcohol - Mean	13.1341	12.2405	13.7115
3	Ash - Mean	2.41765	2.24636	2.45377
4	Ash_Alcanity - Mean	21.2412	20.1909	17.282
10	Color_Intensity - Me...	7.23471	3.01894	5.44459
7	Flavanoids - Mean	0.818824	2.09591	2.96918
11	Hue - Mean	0.691961	1.0607	1.0677
5	Magnesium - Mean	98.6667	93.1364	107.787
2	Malic_Acid - Mean	3.30725	1.8997	1.99705
8	Nonflavanoid_Phen...	0.451961	0.359394	0.28918
12	OD280 - Mean	1.69667	2.81682	3.15475
9	Proanthocyanins - ...	1.14588	1.62788	1.92295
13	Proline - Mean	619.059	509.485	1110.64
14	Silhouette - Mean	0.607639	0.550472	0.609782
6	Total_Phenols - Mean	1.68392	2.26182	2.84213

Figura 47. Características de cada *cluster* en Orange.

Finalmente, podemos comprobar de forma visual la calidad de los *clusters* en el nodo *Silhouette Plot*:

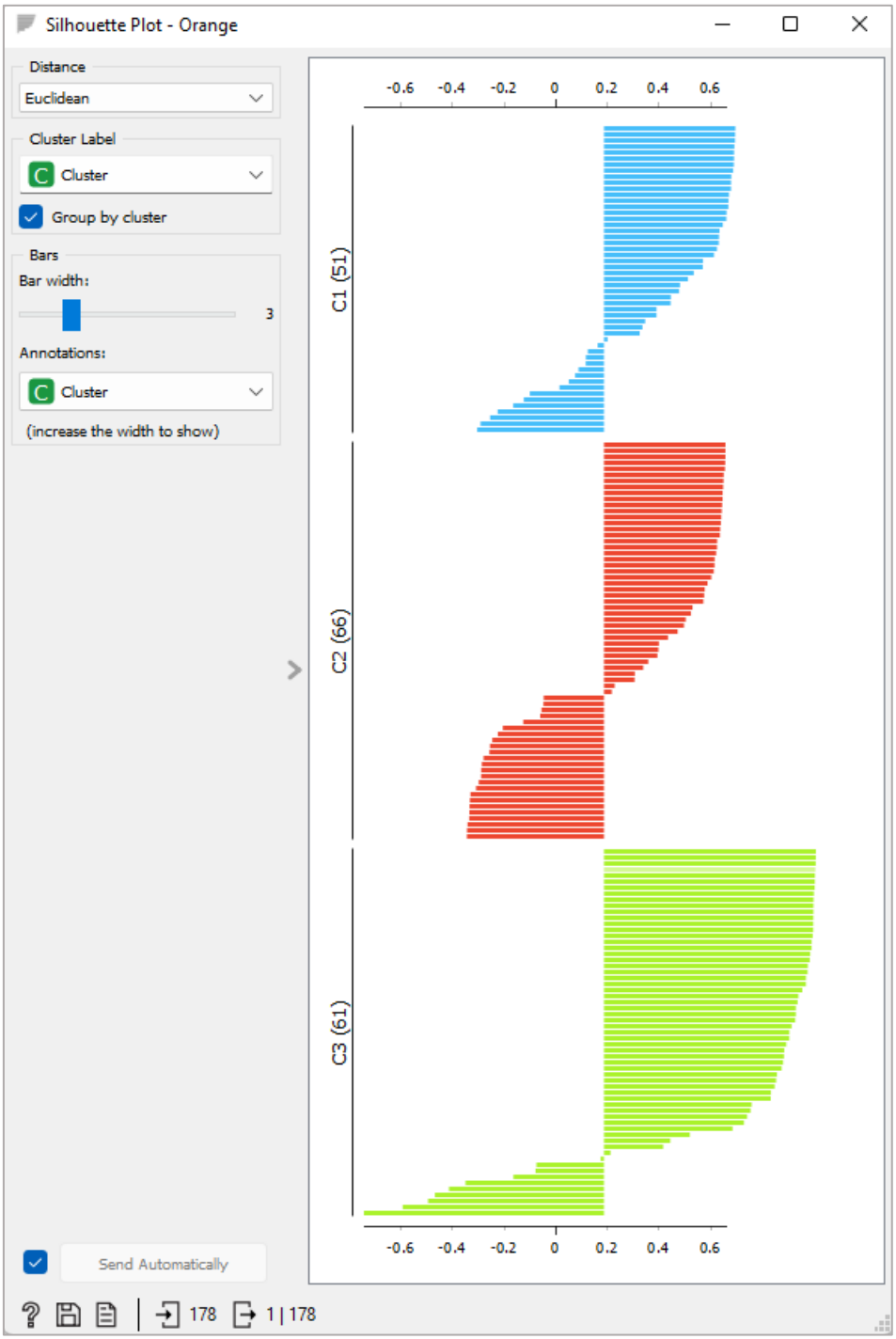


Figura 48. *Silhouete Plot* representado directamente en Orange.

Observando que las puntuaciones más altas se obtienen en el *cluster* 3, es decir, que sus puntos, en general, se encuentran muy cercanos al centro del *cluster*, pese a que a su vez tiene los puntos más alejados (puntuaciones más negativas).

7.3. Tratamiento con Knime

Proceso definido inicialmente:

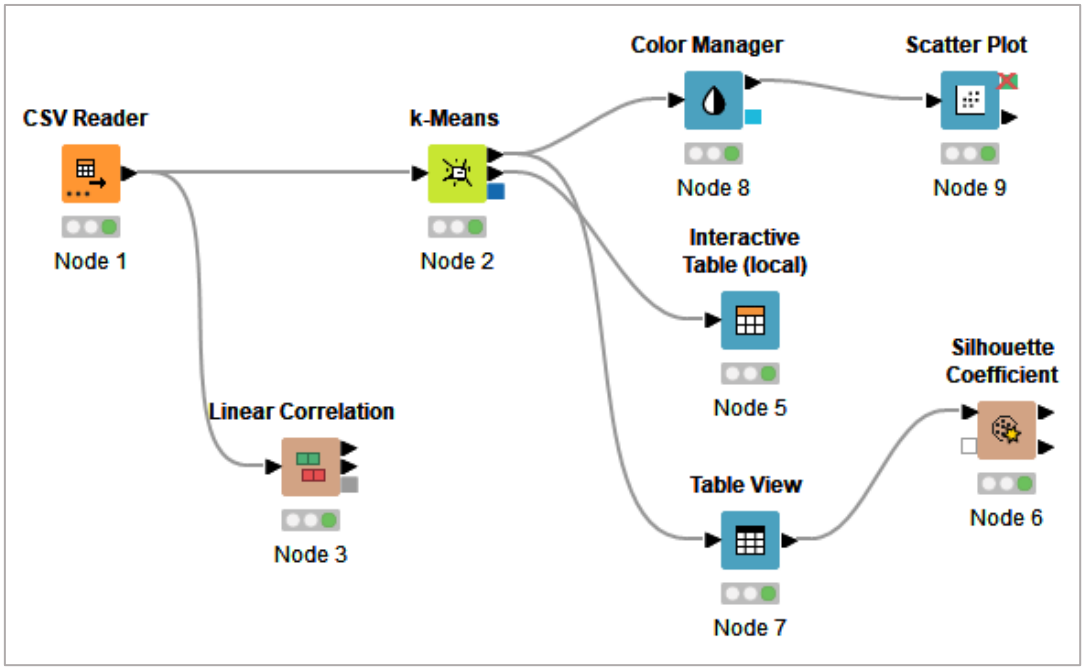


Figura 49. Proceso inicial construido en Knime.

Configurando el nodo *k-Means* para que divida los datos en 3 *clusters*, se puede comprobar la cantidad de instancias que se le asignan a cada uno:

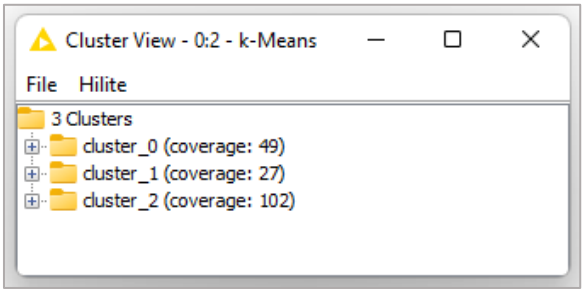


Figura 50. Cantidad inicial de instancias por *cluster* en Knime.

Se comprueba la disparidad en el reparto respecto al conjunto original, por lo que se decide Normalizar los datos para que todas las variables se encuentren comprendidas entre 0 y 1, quedando el proceso como sigue:

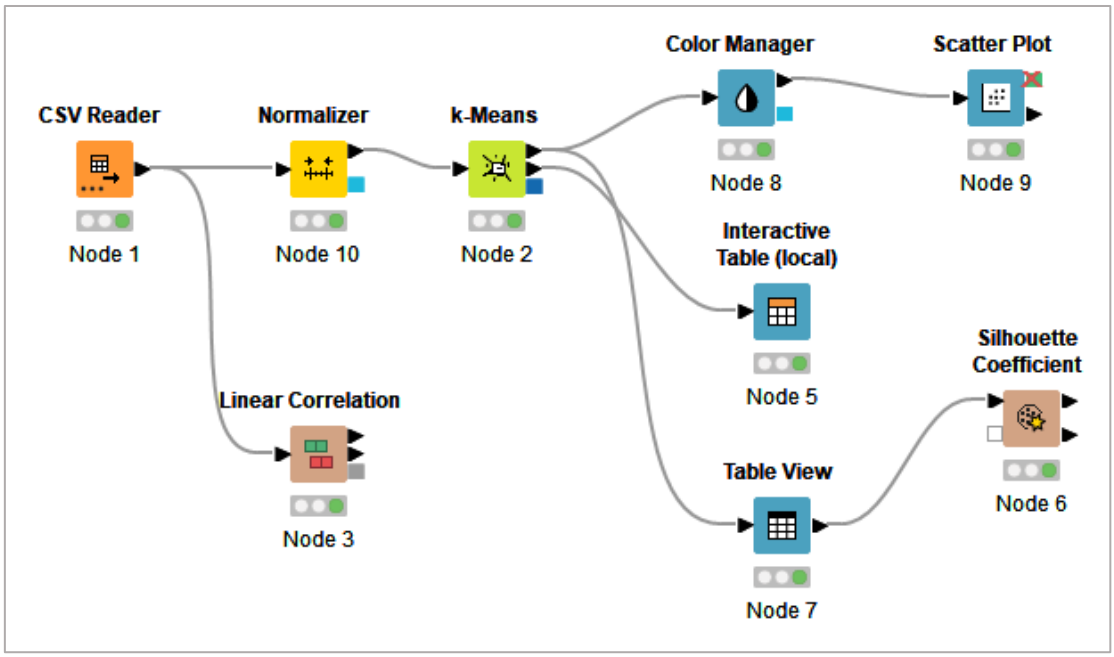


Figura 51. Proceso construido en Knime tras añadir un nodo de normalización.

Comprobando que ya se consigue una mayor aproximación:

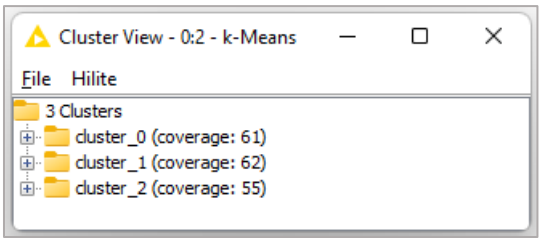


Figura 52. Cantidad de instancias por *cluster* en Knime tras la normalización de variables.

En *Scatter Plot* se representan las mismas variables en los ejes que en Orange para comprobar que la división de los *clusters* es casi idéntica a la conseguida en el software anterior:

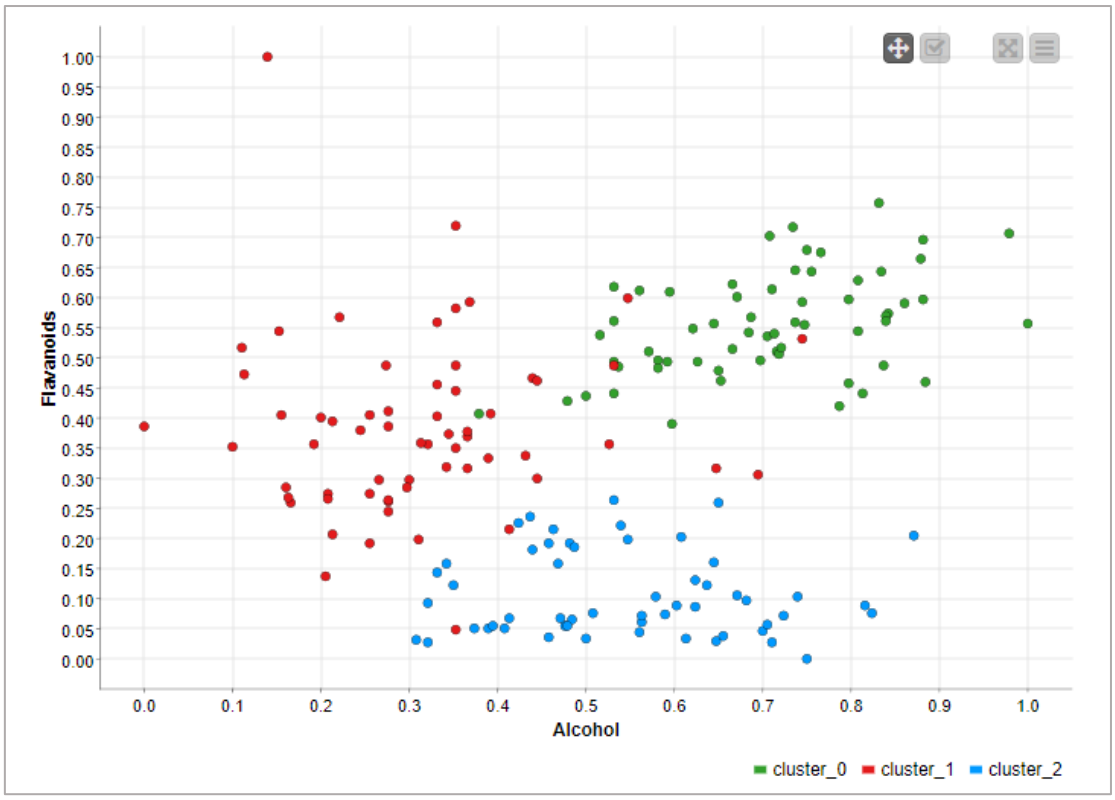


Figura 53. Nube de puntos representando 2 variables y los *clusters* en Knime.

En el nodo *Interactive Table* se muestran directamente las características de cada *cluster* según las medias de los atributos de las instancias que se han asignado a cada uno:

Table View - 0:5 - Interactive Table (local)

File Edit Hilite Navigation View

Row ID	Alcohol	Malic_Acid	Ash	Ash_Al...	Magnes...	Total_P...	Flavan...	Nonflav...	Proant...	Color_I...	Hue	OD280	Proline
cluster_0	0.706	0.248	0.585	0.344	0.411	0.642	0.555	0.3	0.477	0.355	0.478	0.69	0.594
cluster_1	0.311	0.237	0.473	0.5	0.248	0.453	0.382	0.412	0.397	0.148	0.474	0.589	0.156
cluster_2	0.545	0.478	0.56	0.538	0.311	0.245	0.107	0.619	0.228	0.483	0.193	0.161	0.247

Figura 54. Características de los *clusters* en Knime.

Silhouette Coefficient tiene una salida con una tabla para mostrar la media de puntuación de cada uno de los *clusters*, comprobando que el cluster_0, el que aparece coloreado en verde en el gráfico superior, es el que logra una puntuación mayor, como sucedía también en Orange:

Row ID	Mean Silhouette Coefficient
cluster_0	0.374
cluster_1	0.193
cluster_2	0.341
Overall	0.301

Figura 55. Media de la métrica de silueta conseguida en cada *cluster* en Knime.

7.4. Tratamiento con RapidMiner

Proceso definido para su aplicación práctica en RapidMiner:

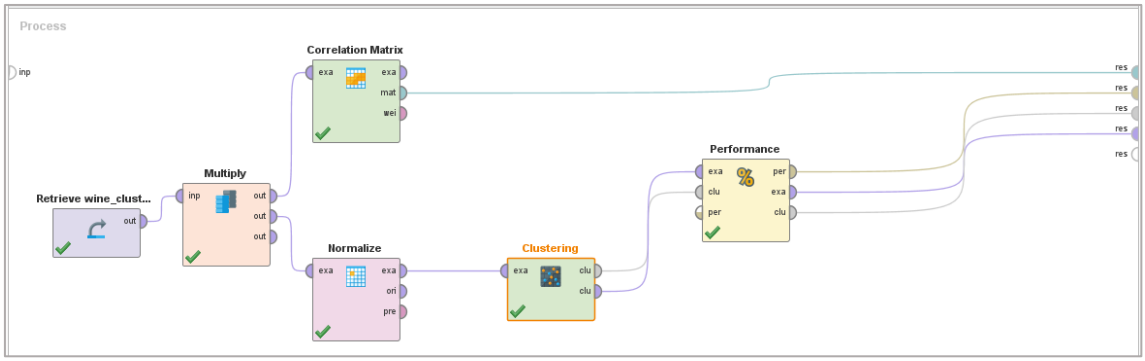


Figura 56. Proceso seguido en RapidMiner para crear el modelo de *clustering*.

Inicialmente se implementa el proceso sin normalizar los datos de entrada al nodo *Clustering*, logrando unos resultados que, al igual que en Knime, distan del reparto original de los 3 tipos de vino:

Cluster Model

```
Cluster 0: 102 items
Cluster 1: 27 items
Cluster 2: 49 items
Total number of items: 178
```

Figura 57. Cantidad inicial de instancias por *cluster* en RapidMiner.

Tras normalizar los datos ya se reparten las instancias en los 3 *clusters* de una forma similar al resto de programas:

Cluster Model

```
Cluster 0: 49 items
Cluster 1: 67 items
Cluster 2: 62 items
Total number of items: 178
```

Figura 58. Cantidad de instancias por *cluster* en RapidMiner tras normalizar variables.

En este software se puede mostrar de una forma gráfica las características de cada *cluster*:

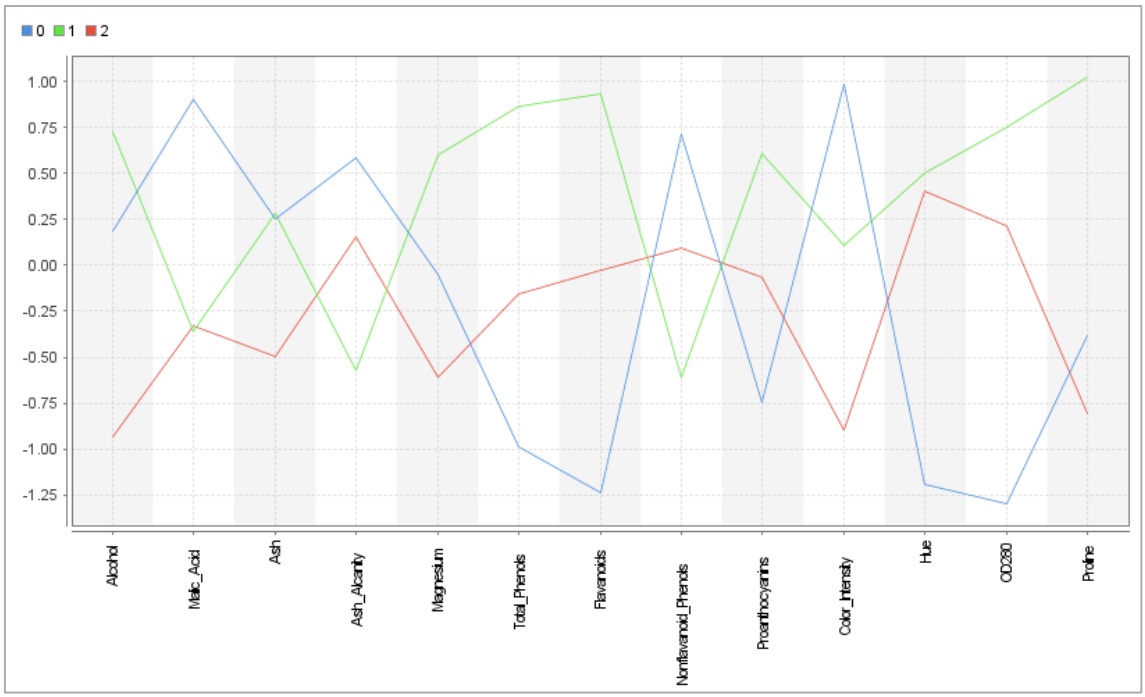


Figura 59. Gráfico con las características de cada *cluster* en RapidMiner.

Además de representar una nube de puntos con las variables de *Flavanoids* y *Alcohol* en los ejes cómo se reparten los 3 *clusters,* observando que en este software existe alguna diferencia más que en el resto de los softwares, como se puede apreciar, por ejemplo, en varias observaciones que el modelo asigna al cluster_1 (azul) pese a estar más cercanas al cluster_2 (verde):

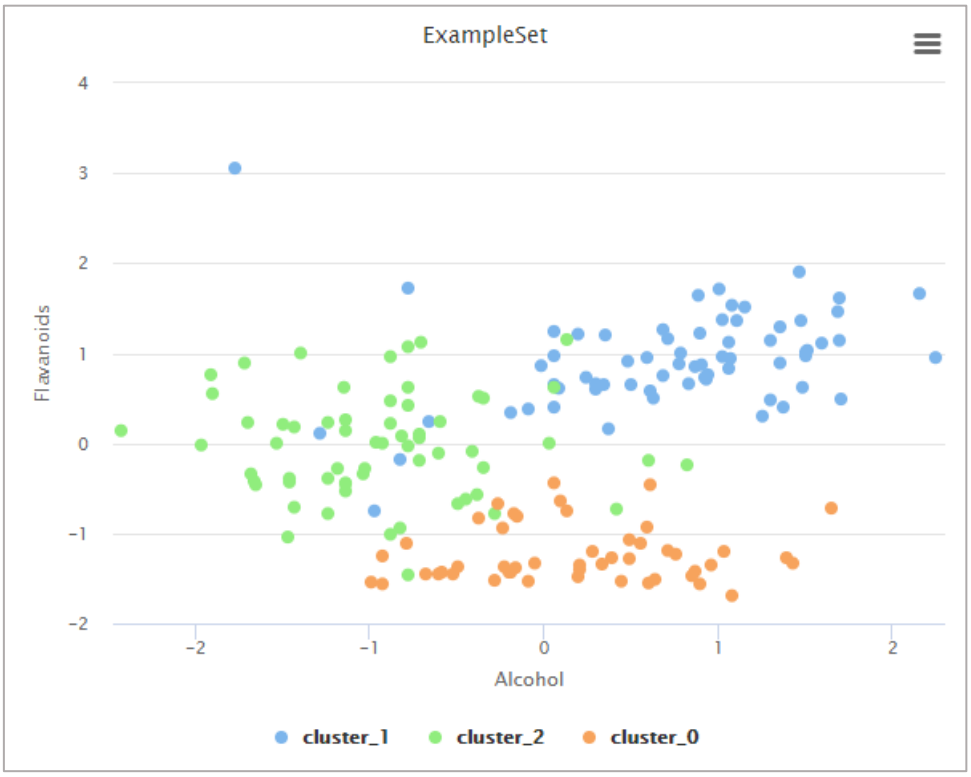


Figura 60. Representación en nube de puntos de dos variables y los *clusters* en RapidMiner.

En este software no se encuentra implementada la puntuación de silueta. Lo más parecido que se obtiene para determinar la calidad de los *clusters* son las distancias medias de las instancias a cada centroide del *cluster,* ofrecido por una de las salidas del nodo *Performance,* comprobando que según esta métrica es el cluster_0 (el coloreado anteriormente en naranja) el que cuenta con un mejor resultado:

PerformanceVector

```
PerformanceVector:
Avg. within centroid distance: -7.164
Avg. within centroid distance_cluster_0: -6.183
Avg. within centroid distance_cluster_1: -7.081
Avg. within centroid distance_cluster_2: -8.030
Davies Bouldin: -1.393
```

Figura 61. Salida de RapidMiner con las distancias medias a cada centroide.

7.5. Tratamiento con Weka

Se selecciona *SimpleKMeans* en la pestaña *Cluster* y se le indica de forma manual que el número de *clusters* debe ser 3:

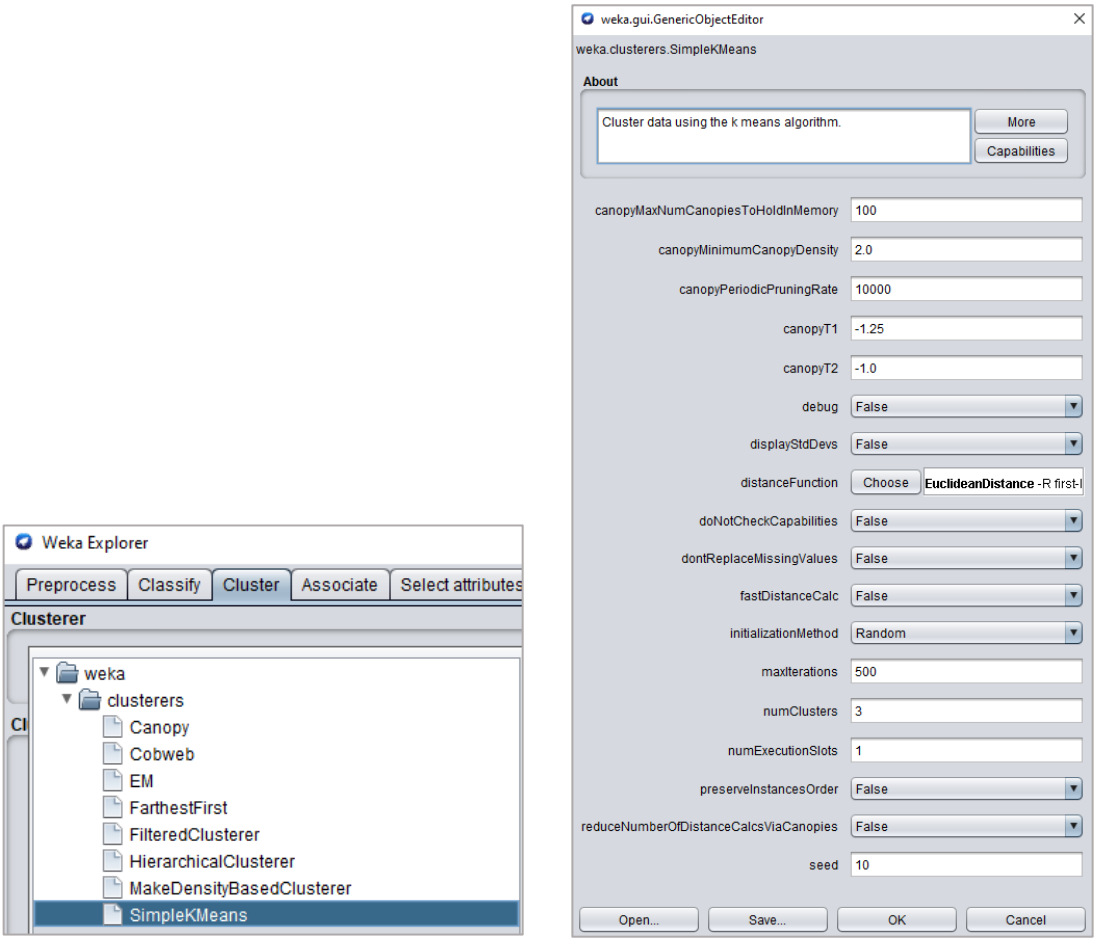


Figura 62. Tipo de algoritmo seleccionado y ventana con su configuración en Weka.

Pudiendo observar en la salida el número de vinos que componen cada *cluster*:

```
Clustered Instances

0       60 ( 34%)
1       55 ( 31%)
2       63 ( 35%)
```

Figura 63. Instancias por *cluster* en Weka

Visualizando cómo, por ejemplo, los *clusters* quedan divididos con claridad al representar *Alcohol* en el eje X y *Flavanoids* en el eje Y:

Figura 64. Nube de puntos representando 2 variables y los *clusters* en Weka.

Logrando una división muy similar a las obtenidas en Orange y Knime.

La salida también muestra los centroides de cada *cluster* según los atributos de los vinos que pertenecen a cada uno:

Final cluster centroids:

Attribute	Full Data (178.0)	Cluster# 0 (60.0)	1 (55.0)	2 (63.0)
Alcohol	13.0006	13.7193	13.0998	12.2295
Malic_Acid	2.3363	1.964	3.1609	1.9711
Ash	2.3665	2.4565	2.4075	2.2451
Ash_Alcanity	19.4949	17.2783	21.0436	20.254
Magnesium	99.7416	107.8667	98.6545	92.9524
Total_Phenols	2.2951	2.8455	1.6898	2.2994
Flavanoids	2.0293	2.9748	0.8478	2.1602
Nonflavanoid_Phenols	0.3619	0.2887	0.4578	0.3478
Proanthocyanins	1.5909	1.9273	1.1336	1.6697
Color_Intensity	5.0581	5.4627	6.9365	3.0329
Hue	0.9574	1.0718	0.7168	1.0586
OD280	2.6117	3.1573	1.7093	2.8798
Proline	746.8933	1117.8167	624.8545	500.1746

Figura 65. Características de cada *cluster* en Weka.

Una vez realizada la agrupación en los 4 softwares, se comprueba en qué atributos destaca cada uno de los *clusters* formados teniendo en cuenta el valor medio de los vinos que los forman, renombrándolos de la siguiente forma para que coincidan, ya que cada programa les asigna un nombre y color distintos:

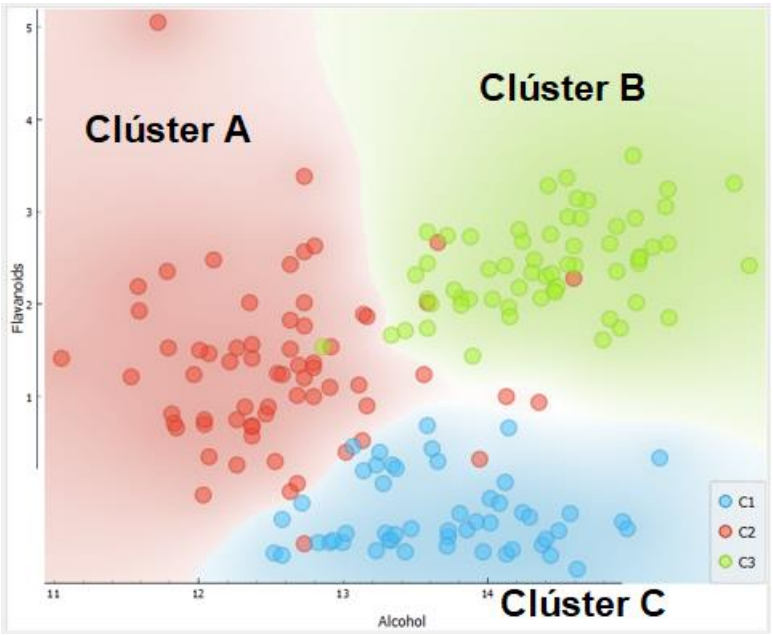


Figura 66. *Clusters* representados en Orange con nombre unificado para presentar resultados.

Destacando en valores máximos y mínimos en los siguientes atributos:

Clúster	Valor	Orange	Knime	RapidMiner	Weka
Clúster A	**Máx.**				
	Mín.	Alcohol Color_Intensity Ash Magnesium	Alcohol Color_Intensity Ash Magnesium	Alcohol Color_Intensity Ash Magnesium	Alcohol Color_Intensity Ash Magnesium
Clúster B	**Máx.**	Alcohol Total_Phenols Flavanoids Proline	Alcohol Total_Phenols Flavanoids Proline	Alcohol Total_Phenols Flavanoids Proline	Alcohol Total_Phenols Flavanoids Proline
	Mín.	Nonflavanoid_Phenols Ash_Alcanity	Nonflavanoid_Phenols Ash_Alcanity	Nonflavanoid_Phenols Ash_Alcanity	Nonflavanoid_Phenols Ash_Alcanity
Clúster C	**Máx.**	Malic_Acid Ash_Alcanity Nonflavanoid_Phenols Color_Intensity	Malic_Acid Ash_Alcanity Nonflavanoid_Phenols Color_Intensity	Malic_Acid Ash_Alcanity Nonflavanoid_Phenols Color_Intensity	Malic_Acid Ash_Alcanity Nonflavanoid_Phenols Color_Intensity
	Mín.	Total_Phenols Flavanoids Hue OD280	Total_Phenols Flavanoids Hue OD280	Total_Phenols Flavanoids Hue OD280	Total_Phenols Flavanoids Hue OD280

Figura 67. Tabla con los máximos y mínimos de los 3 *clusters* en cada software.

Concluyendo lo siguiente:

- Los *clusters* creados en los 4 programas **coinciden en sus características** atendiendo a los valores medios máximos y mínimos en los que destacan los vinos que los componen, aunque haya discrepancias en la clasificación de alguno de ellos.
- Como curiosidad, se aprecia que en el *Clúster A* la media de los atributos de sus vinos no alcanza en ningún caso el valor máximo de los 3 *clusters*, como ya se visualizaba en el gráfico de líneas de la *Figura 59* representado en RapidMiner.

8. COMPARATIVA DE LOS SOFTWARES

En este apartado se realiza una comparativa tras el uso y aplicación de las técnicas estudiadas en los 4 programas, comenzando por los resultados obtenidos en los algoritmos implementados y terminando por una comparativa más cualitativa sobre la experiencia de uso en cada uno de ellos.

8.1. Resultados de los algoritmos

8.1.1. Aprendizaje Supervisado

Tabla-resumen de los resultados en el ejemplo de clasificación:

Software	Tipo Validación	Exactitud	Área AUC
Orange	Validación Cruzada	77,94%	0,839
Knime	Conjuntos de entrenamiento y test	**78,27%**	**0,843**
RapidMiner	Conjuntos de entrenamiento y test	**76,71%**	**0,817**
Weka	Validación Cruzada	78,20%	0,831

Los resultados son muy similares, sin diferencias significativas:

- El mejor resultado se alcanza en Knime y el peor en RapidMiner, en ambos programas se utilizó la validación dividiendo el conjunto de datos en entrenamiento y test.
- Se aprecia cómo la exactitud o *accuracy* (métrica que se considera como la ideal en este ejemplo) está directamente relacionada con el área AUC (área bajo la curva ROC).

8.1.2. Aprendizaje No Supervisado

En esta ocasión se compara la distribución de las instancias del conjunto de datos en los diferentes *clusters*, debido a que no se dispone de la métrica de silueta en todos los softwares:

Software	C1	C2	C3
Orange	51	66	61
Knime	61	62	55
RapidMiner	67	62	49
Weka	60	63	55
Original	**59**	**71**	**48**

La distribución de los vinos en los 3 *clusters* es bastante homogénea en los 4 casos, siendo en 3 de los 4 programas (excepto en RapidMiner) en los que al *cluster 2* se le asignan el máximo número de instancias.

Como se comentó, en las técnicas de aprendizaje no supervisado no existe una métrica que determine si un modelo se desempeña mejor o peor, será la experiencia y el conocimiento del contexto/negocio de quien lo desarrolla lo que decida si tiene sentido o no.

8.2. Usabilidad y experiencia de uso

En este apartado se destaca lo mejor y lo peor de cada programa sobre la experiencia personal de uso tras aplicar las distintas técnicas descritas anteriormente.

8.2.1. Orange

A favor:

- Cuenta con una documentación muy completa y vistosa en su web, además de un blog y ejemplos de uso, lo que provoca que facilita el aprendizaje de la herramienta.
- Sencillez para conectar los distintos operadores, lo que facilita una rápida curva de aprendizaje.
- Facilidad para implementar con el nodo *Rank* un ranking de variables según la información que aportan al modelo, ideal para reducir y seleccionar el número de variables.
- Facilita por defecto la métrica *Silhouette Score* para conocer el número óptimo de clústeres en un aprendizaje no supervisado.
- Representación directa de la curva ROC.
- Estadísticas descriptivas y correlaciones entre las variables muy fácil de representar.
- Posibilidad de comprimir los nodos del árbol de decisión para simplificar su representación.
- No se aprecia sensibilidad a la normalización de los datos a la hora de realizar el *clustering* con *k-means*, lo que hace más sencilla su implementación, al menos, en el ejemplo presentado en el documento.

En contra:

- Para presentar las características de los *cluster* es necesario la combinación de varios nodos mientras que en otros de los softwares se puede visualizar de una forma más directa.

8.2.2. Knime

A favor:

- Numerosos recursos propios de aprendizaje y ayuda, tanto destinados a dar los primeros pasos en el programa como para realizar consultas con un nivel mayor de detalle cuando ya se dominan los aspectos básicos.
- *Cheat sheets* (“chuletas”) muy útiles (https://www.knime.com/cheat-sheets), muy socorridas en los entornos de software y programación, que se pueden utilizar como guía rápida de referencia según diferentes categorías.
- Facilidad para realizar un análisis exploratorio inicial:
    - Resumen estadístico muy completo con tan solo añadir un nodo.

    - Nodo dedicado a representar la matriz de correlaciones entre variables.
- Posibilidad de comprimir los nodos del árbol de decisión para simplificar su representación.
- Rapidez a la hora de mostrar en una tabla las características de los *clusters*.

En contra:

- Complejidad para representar el gráfico de *Silhouette*, ya que no se encuentra implementado directamente, siendo necesaria la combinación de varios nodos, mientras que con otros softwares su representación es directa.
- Sensible a la normalización de los datos antes de realizar un *clustering* con *k-means*.

8.2.3. RapidMiner

A favor:

- Con *Auto Model* el programa te guía en el modelo que deseas construir, ayudándote en la selección y normalizado de variables, una funcionalidad que no se presenta en el resto de los programas de este documento.
- Se puede encontrar una amplia documentación sobre todos los operadores del programa, además de contar con una comunidad de usuarios que ayuda a resolver dudas en un foro.
- Es posible representar de forma directa la curva ROC de un clasificador.
- Se obtiene de forma sencilla los pesos que les otorga un árbol de decisión a cada variable, útil para conocer la importancia de cada una.
- Facilidad para especificar qué salidas quieres obtener del proceso, independientemente del paso en el que se encuentren.
- Simplicidad en la representación gráfica de las características de cada *cluster*.

En contra:

- Difícil determinar el número óptimo de *clusters*, no tiene implementada ninguna funcionalidad para resolverlo directamente.
- No tiene implementada la métrica de *silhouette*.
- Complica la representación de árboles de decisión complejos.
- Es el software en el que se han experimentado los mayores tiempos de carga y ejecución.

8.2.4. Weka

A favor:

- Facilidad para obtener representaciones visuales y estadísticos básicos con solo cargar el conjunto de datos.
- No se aprecia sensibilidad a la normalización de los datos a la hora de realizar el *clustering* con *k-means*, lo que hace más sencilla su implementación.
- La salida con los resultados del clasificador ofrece una información muy completa con las métricas suficientes para evaluar las bondades del modelo creado.

En contra:

- En su aplicación *Explorer*, al ser el programa más antiguo de los 4 estudiados, la interfaz que presenta es la más desfasada visualmente.
- Único programa de los estudiados que no representa el flujo de datos mediante un proceso visual.
- Documentación online escasa, teniendo que recurrir en ocasiones a la descarga de documentos de la web para encontrar información.
- No tiene implementada la métrica de silueta para valorar la "calidad" de los *clusters*.

9. CONCLUSIONES

La elección de utilizar un software u otro se debe decantar más por la variedad de visualizaciones que presente, que ayuden en el análisis exploratorio de los datos, la cantidad de modelos a implementar y la facilidad de uso que por sus resultados tras la aplicación de sus algoritmos, ya que las diferencias en las distintas métricas de evaluación en el problema de clasificación presentado apenas distan unos de otros, sucediendo lo mismo en el caso de agrupación, en el que tampoco existen diferencias significativas.

Las razones de esas pequeñas diferencias son la aleatoriedad de algunos de los procesos (o en las particiones de los datos) o pequeñas modificaciones en la configuración de los algoritmos.

Es por ello que, según la experiencia presentada, el software que aventaja al resto es **Orange** esencialmente por los siguientes motivos:

- Interfaz muy vistosa.
- Curva de aprendizaje rápida, siendo el más intuitivo.
- Gran cantidad de documentación y contenidos de ayuda.
- Variedad de *widgets*, tanto de visualizaciones, modelos y métricas.

En el otro extremo se encuentra **Weka**, al utilizar su aplicación *Explorer*, con un aspecto más desfasado y una interfaz mucho menos visual que los

otros 3, donde no quedan tan claros los pasos intermedios del proceso, sobre todo en las transformaciones intermedias de los datos, filtros, agrupamientos, etc.

De **Knime** se puede destacar que es el más similar a Orange, sin llegar a ser tan completo y sencillo en algunos aspectos, y de **RapidMiner** que sus principales virtudes se encuentran en la facilidad para obtener las salidas deseadas y su funcionalidad de crear un modelo guiado, ideal para construir de forma rápida un modelo que puede servir como base para mejorarlo posteriormente de forma manual por el usuario.

10. REFERENCIAS

- Altalhi, A. H., Luna, J. M., Vallejo, M. A., & Ventura, S. (2017). Evaluation and comparison of open source software suites for data mining and knowledge discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7* (3), e1204.
- Fawcett, T., Provost, F. (1997) Adaptive fraud detection. *Data mining and knowledge discovery*, 1 (3), p. 291-316.
- Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking.* O'Reilly Media, Inc.
- Ramírez, C. F., Orallo, J. H., & Ramírez Quintana, M. J. (2003). *Introducción A La Minería De Datos*.
- Riquelme Santos, J. C., Ruiz, R., & Gilbert, K. (2006). Minería de datos: Conceptos y tendencias*. Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial*, 10 *(29), 11-18*.
- Shearer, C. (2000*)* The CRISP-DM Model: The New Blueprint for Data Mining*. Journal of Data Warehousing*, 5, 13-22.
- Thabtah, F., Hammoud, S., Kamalov, F., & Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences, 513*, 429-441.