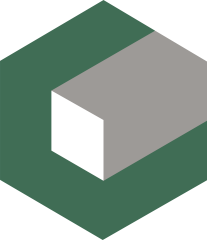
**Universidad Tecnológica de La Habana**

**José Antonio Echeverría**



**CUJAE**

**Facultad de Ingeniería Informática**

Investigación de Prácticas 2

**“Desarrollo de Componente AutoML para tareas de Agrupamiento en Knime”**

**Autor:  
Mario Perera Valdes**

**Tutor:  
Dr. C. Raisa Socorro Llanes**

Resumen

El preprocesamiento de datos es una fase crucial en el aprendizaje automático, ya que permite preparar los datos de entrada para ser utilizados en algoritmos de agrupamiento y así dar con insights de mayor relevancia. Sin embargo, el preprocesamiento de datos puede ser un proceso complejo y costoso en tiempo y recursos.

Por lo tanto, este trabajo se basa en el desarrollo de la versión beta de un componente en Knime encargado de automatizar este preprocesamiento y aprendizaje automático, con el objetivo de que el producto llegue a demostrar una eficiencia y eficacia muy favorable para las tareas de agrupamiento.

El objetivo de este trabajo es construir un componente de AutoML en Knime para tareas de Agrupamiento.

Palabras claves: AutoML, pre-procesado, KNIME, componente, insights .

**Abstract**

Data preprocessing is a crucial phase in machine learning, as it allows input data to be prepared to be used in clustering algorithms and thus provide more relevant insights. However, data preprocessing can be a complex process and costly in time and resources.

Therefore, this work is based on the development of the beta version of a component in Knime responsible for automating this preprocessing and machine learning, with the aim of the product demonstrating very favorable efficiency and effectiveness for grouping tasks.

The objective of this work is to build an AutoML component in Knime for Clustering tasks.

Keywords: AutoML, preprocessing, KNIME, component.

Tabla de contenido

Contenido

[Introducción 1](#_Toc176688031)

[Capítulo 1: Fundamentos Teóricos 4](#_Toc176688032)

[1.1 Minería de Datos 4](#_Toc176688033)

[1.2 Machine Learning 5](#_Toc176688034)

[1.3 Algoritmos de Agrupamiento 6](#_Toc176688035)

[1.3.1 Agrupamiento por distancias o centroides: 6](#_Toc176688036)

[1.3.2 Agrupamiento por Densidad 7](#_Toc176688037)

[1.3.3 Agrupamiento por Jerarquía 8](#_Toc176688038)

[1.4 AutoML 9](#_Toc176688039)

[1.4.1 Pre-procesado 10](#_Toc176688040)

[1.4.2 Procesado (Optimización de Hiperparametros) 11](#_Toc176688041)

[1.4.3 Representación de Resultados 12](#_Toc176688042)

[1.5 KNIME 13](#_Toc176688043)

[Conclusiones parciales 15](#_Toc176688044)

[Capítulo 2: Desarrollo del Componente AutoML Clustering 16](#_Toc176688045)

[2.1 Mecanismo de pre-procesado 16](#_Toc176688046)

[2.1.1 Recolección de datos 16](#_Toc176688047)

[2.1.2 Pre-procesado de datos tipo string 17](#_Toc176688048)

[2.1.3 Manejo de valores faltantes 18](#_Toc176688049)

[2.1.4 Codificación y Normalización 21](#_Toc176688050)

[2.2 Prototipo del Componente AutoML Clustering 23](#_Toc176688051)

[2.2.1 Requisitos del Componente AutoML Clustering 24](#_Toc176688052)

[2.3 Optimización y Procesamiento de K-Means 26](#_Toc176688053)

[2.4 Optimización y Procesamiento de DBSCAN 27](#_Toc176688054)

[2.5 Optimización y Procesamiento de Hierarchical 28](#_Toc176688055)

[2.6 Post-Procesamiento, puntuación y representación de los resultados 29](#_Toc176688056)

[2.6.1 Puntuación de los resultados(Componente Scorer) 29](#_Toc176688057)

[2.6.2 Representación de los resultados(Componente Scorer) 31](#_Toc176688058)

[Bibliografía 34](#_Toc176688059)

**Índice de Figuras**

[Figura 1 Representación de K-means 7](#_Toc176688060)

[Figura 2 Aplicación de DBSCAN y sus Conceptos claves 8](#_Toc176688061)

[Figura 3 Dendograma como método de representación las relaciones Jerárquicas 9](#_Toc176688062)

[Figura 4 Ejemplo de flujo KNIME 14](#_Toc176688063)

[Figura 5 Recolección de Datos 17](#_Toc176688064)

[Figura 6 Vista previa de flujo KNIME para el filtrado de valores únicos 18](#_Toc176688065)

[Figura 7 Vista previa de flujo KNIME para el reemplazo por ’other’ 18](#_Toc176688066)

[Figura 8 Diagrama de flujo para la imputación de valores faltantes 19](#_Toc176688067)

[Figura 9 Vista previa de flujo KNIME para kNNI 20](#_Toc176688068)

[Figura 10 Vista previa de flujo KNIME para kMI 20](#_Toc176688069)

[Figura 11 Proceso de Normalización 21](#_Toc176688070)

[Figura 12 Vista previa de flujo KNIME para la codificación One-Hot 22](#_Toc176688071)

[Figura 13 Diagrama de flujo para la normalización 22](#_Toc176688072)

[Figura 14 Componente AutoML Clustering 24](#_Toc176688073)

[Figura 15 Diagrama de flujo general del componente AutoML Clustering 25](#_Toc176688074)

[Figura 16 Diagrama de flujo de la optimización de hiperparámetros 26](#_Toc176688075)

[Figura 17 Diagrama de flujo para la optimización de hiperparámetros de K-Means 27](#_Toc176688076)

[Figura 18 Diagrama de flujo de procesamiento de K-Means 27](#_Toc176688077)

[Figura 19 Diagrama de flujo para la optimización de hiperparámetros de DBSCAN 28](#_Toc176688078)

[Figura 20 Diagrama de flujo de procesamiento de DBSCAN 28](#_Toc176688079)

[Figura 21 Diagrama de flujo para la optimización de hiperparámetros de Hierarchical 29](#_Toc176688080)

[Figura 22 Diagrama de flujo para el procesamiento de Hierarchical 29](#_Toc176688081)

[Figura 23 Funcionamiento de Scorer 30](#_Toc176688082)

[Figura 24 Representación de Metricas 30](#_Toc176688083)

[Figura 25 Parametros de Entrada encargados de la representación 31](#_Toc176688084)

[Figura 26 Representación de Resultados 32](#_Toc176688085)

**Índice de Tablas**

[Tabla 1 Asociación de los objetivos de AutoML con las etapas de KDD Fases de KDD 5](#_Toc176688103)

[Tabla 2 Técnicas de Pre-procesamiento en tareas de agrupamiento 11](#_Toc176688104)

# Introducción

En la actualidad, en el mundo se generan cada vez más datos y se almacenan en diversos tipos de sistemas, lo que nos presenta un gran desafío: ¿cómo convertir estos datos en información valiosa que pueda ser utilizada para tomar decisiones basadas en la lógica? Para resolver este reto, se han desarrollado diversas técnicas y herramientas para extraer información útil de grandes cantidades de datos.

El proceso de descubrimiento de conocimiento en bases de datos (KDD, por sus siglas en inglés), es un procedimiento que se utiliza para extraer conocimiento útil y relevante, a partir de grandes cantidades de datos almacenados en diversos sistemas (Hernández Orallo et al., 2004). Este consta de varias fases: integración y recopilación; selección, limpieza y transformación; aplicación de algoritmos de minería de datos; evaluación e interpretación; así como la difusión y uso del conocimiento obtenido (Han et al., 2011). El proceso de descubrimiento de conocimiento en bases de datos ha sentado las bases para una disciplina relacionada, conocida como minería de datos.

La minería de datos es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos (Hernández Orallo et al., 2004). Para la realización de esta actividad el ser humano necesita auxiliarse en la capacidad de procesamiento de las computadoras, debido a los grandes volúmenes de datos que se usan en situaciones reales. Por esta razón aparecen los algoritmos Machine Learning.

Machine learning es un subconjunto de la inteligencia artificial (IA). Se enfoca en enseñar a las computadoras para que aprendan de los datos y mejoren con la experiencia en lugar de ser explícitamente programadas para hacerlo. Esta es una de las áreas de estudio de la Informática, acaparadora de una gran cantidad de focos en la actualidad; tanto por el sector académico como por el empresarial. Para comprender grandes volúmenes de datos y encontrar relaciones ocultas entre ellos, existe una tarea en particular llamada agrupamiento.

Esta tarea es fundamental dentro del machine learning. Es la disciplina, enmarcada en la inteligencia artificial que permite a las máquinas aprender de forma automática, realizando tareas de manera independiente sin necesidad de ser programadas explícitamente para cada situación. Dentro del aprendizaje automático, se encuentran varios grupos de algoritmos, incluyendo los algoritmos de agrupación, que se utilizan para categorizar elementos similares en grupos o clases basándose en sus características comunes [3].

Los algoritmos de agrupación son especialmente útiles en situaciones donde se busca identificar patrones o relaciones inherentes en los datos sin conocer previamente ninguna etiqueta o categoría. Este tipo de algoritmos se aplica en diversas áreas, desde la segmentación de clientes en marketing hasta la detección de anomalías en sistemas de seguridad informática, permitiendo a las organizaciones tomar decisiones más informadas y eficientes basadas en patrones emergentes de los datos [4].

Este trabajo se enfoca en particular en el desarrollo del aprendizaje Automático Automatizado (AutoML) disciplina que nace sobre las bases de *Machine Learning* en procesos de minería de datos. Ha sido desarrollado como una solución para simplificar y acelerar este proceso. AutoML tiene como objetivo tomar estas decisiones de una manera automatizada, objetiva y basada en datos: el usuario simplemente proporciona datos y el sistema AutoML determina automáticamente el enfoque que funciona mejor para esta aplicación en particular (Hutter et al., 2019). Para entender mejor el porque es necesario el AutoML debemos analizar una importante fase del KDD, el preprocesamiento de la información.

El preprocesamiento es una fase crucial que prepara los datos para su análisis posterior. Esta fase es fundamental para garantizar que los algoritmos de minería de datos funcionen de manera efectiva y produzcan resultados precisos. Esta fase suele ser la que más tiempo y esfuerzo toma de los profesionales de esta área. Sin embargo, en los últimos años han aparecido avances en las numerosas técnicas de preprocesamiento, por lo que existe la posibilidad de automatizar parcialmente esta tarea para conseguir los mismos resultados en menos tiempo [15].

Por lo que el propósito del AutoML en la minería de datos y lo que se pretende con este trabajo es facilitar el uso, acceso y desarrollo de los ciclos de vida del procesamiento de información para tareas de agrupamiento.

Una de las herramientas más populares de Minería de Datos es KNIME, precisamente por su baja curva de aprendizaje, enfoque en el pre-procesado y sustitución de la codificación clásica por nodos altamente visuales [24]. Estas características son idóneas para la implementación y uso de funcionalidades de AutoML.

La **situación problemática** es desarrollar el AutoML con Knime.

Relacionado a esta situación se tienen en cuenta precedentes como el componente “AutoML Clasificación (pre-procesado)” desarrollado por Ernesto Carrazana en su tesis de pregrado [23] y se tiene como la mejor referencia la librería “Weka” de KNIME como una herramienta con múltiples alternativas para la implementación de AutoML y sobre todo el trabajo para tareas de agrupamiento el **problema** a resolver es la ausencia de un componente AutoML para tareas de agrupamiento [24,26].

Para resolver este problema se recoge la propuesta de un componente de AutoML en Knime que se desenvuelva en tareas de agrupamiento y se enfoque en el pre-procesamiento de los datos que reciba. Que también sea capaz de aplicar los tres enfoques principales del agrupamiento.

Por lo tanto, este contexto nos lleva al **objetivo general: “Desarrollar un componente de AutoML para tareas de Agrupamiento en Knime”**.

Este objetivo general se desglosa en los siguientes **objetivos específicos** y tareas:

* Analizar el estado del arte de Machine Learning, AutoML y KNIME.

1. Identificar las técnicas de Machine Learning en tareas de agrupamiento.
2. Definir las técnicas de AutoML a aplicar en tareas de agrupamiento.
   1. Identificar y seleccionar las técnicas a aplicar en el pre-procesamiento.
   2. Identificar y seleccionar las técnicas a aplicar en la fase de la construcción del modelo.
   3. Identificar y seleccionar las técnicas a aplicar en la fase de representación de los resultados.
3. Definir la implementación de los nuevos componentes en Knime.

* Desarrollar el componente “AutoML Agrupamiento”.

1. Desarrollar la fase de pre-procesamiento interno del componente.
2. Desarrollar los componentes encargados del procesamiento:
   1. K-Means
   2. DBSCAN
   3. Hierarchical
3. Desarrollar la fase de puntuación y selección del modelo de salida.
4. Implementar la fase de representación de los resultados.

Se define como **objeto de estudio**:

* AutoML en tareas de Minería de Datos
* Campo de acción AutoML en KNIME enfocado en el pre-procesado.
* Tareas de Agrupamiento en la minería de datos

Se define como **campo de acción**:

* AutoML en KNIME.

**Valor Práctico:**

Este trabajo forma parte de una ambiciosa investigación de las ventajas que ofrece la incorporación del AutoML en los procesos de minería de datos. El componente a desarrollar promete resultados satisfactorios, que permitan su inclusión en múltiples tareas descriptivas en las problemáticas actuales y futuras que requieran agrupar datos. Con el gran beneficio de una gran reducción en el tiempo de desarrollo de los flujos de datos. Incluso obtener resultados a la altura de un experto en el campo.

# Capítulo 1: Fundamentos Teóricos

Introducción

Este Capítulo expone los principales conceptos de Machine Learning, AutoML y Minería de Datos. Se enfatiza en el análisis de las características de los algoritmos de Agrupamiento. Adicionalmente se presenta una revisión del estado del arte de las principales herramientas y servicios de AutoML, así como la relación de AutoML con la Minería de Datos.

## 1.1 Minería de Datos

La Minería de Datos se define como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos [23].

El objetivo de la Minería de Datos es convertir datos en conocimiento, por lo que su tarea fundamental es encontrar modelos inteligibles a partir de los datos. La Minería de Datos es un proceso que se integra dentro del proceso de Descubrimiento de Conocimiento en Bases de Datos [23].

Descubrimiento de Conocimiento en Bases de Datos (KDD por sus siglas en inglés) es el complejo proceso de obtener modelos o patrones a partir de bases de datos y la evaluación e interpretación de los mismos. Estos sistemas permiten la selección, limpieza, transformación y proyección de los datos. El proceso de Descubrimiento de Conocimiento en Bases de Datos agrupa los procesos de preparación de los datos, Minería de Datos, obtención de patrones y la evaluación, interpretación y visualización de los resultados [23].

Luego del estudio realizado, en la Tabla 2 se identifican las posibles asociaciones de los objetivos de AutoML con las etapas de KDD.

|  |  |
| --- | --- |
| Fases de KDD | Objetivos de AutoML |
| Preparación de los datos | Etapa con gran proyección de apoyo por las técnicas de AutoML, ejemplo la selección de variables, limpieza y transformación de los datos |
| Minería de Datos | La Minería de Datos agrupa las tareas que más esfuerzos aúnan en AutoML, debido a la criticidad de la selección de hiperparámetros en el desempeño de los modelos |
| Obtención de patrones y evaluación | Todas las herramientas y bibliotecas brindan un buen soporte a esta etapa, dando la posibilidad de elegir entre múltiples patrones y métricas |
| Interpretación y visualización | Si bien se presenta cubierto por las herramientas y lenguajes, existe una gran diferencia en la calidad y variedad que ofertan los servicios de las grandes empres |

Tabla Asociación de los objetivos de AutoML con las etapas de KDD Fases de KDD

## 1.2 Machine Learning

Machine Learning, conocido como Aprendizaje Automático en español; es una de las ramas con mayor relevancia en el campo de la Inteligencia Artificial y la Informática en general. El aprendizaje o learning se puede definir como la habilidad del cambio acorde a los estímulos externos, con la posibilidad de analizar las experiencias previas [4]; siguiendo un enfoque diferente, se puede establecer el aprendizaje como la búsqueda de la mejor descripción o conceptualización de los datos en un espacio de búsqueda [5]. Por lo tanto, Machine Learning se conceptualiza como el conjunto de técnicas que priorizan o incrementan la adaptabilidad, empleando conjuntos de datos de entrada de diferente índole [4]. Las computadoras en la actualidad almacenan varios tipos de datos tales como: textos, números, imágenes, audios, videos. Todos ellos son empleados como base para el entrenamiento de los modelos de Machine Learning [2].

**Tipos de aprendizajes:**

∙ Aprendizaje supervisado: proceso que pretende desarrollar un modelo predictivo o inferencial, empleando un conjunto de datos etiquetados [5]. Este tipo de aprendizaje es empleado en los campos de la detección de patrones, detección de bots, procesado de lenguaje natural y análisis de 5 sentimientos [6]. Regresión y clasificación se encuentran entre las técnicas emblemáticas de esta tarea.

∙ Aprendizaje no supervisado: proceso que pretende etiquetar los datos de entradas, con la ausencia de conocimiento previo y retroalimentación [5]. Dos de las técnicas más empleadas son agrupamiento y asociación. Es aplicado en la segmentación de objetos, detección de similitudes y etiquetado automático [7].

∙ Aprendizaje por refuerzo: proceso dinámico, en el que se otorga una recompensa al ejecutar diferentes acciones en los diferentes estados, de esta manera se pretende aprender la política que obtenga la mayor recompensa [5].

∙ Aprendizaje semi-supervisado: proceso híbrido entre el aprendizaje supervisado y no supervisado, pues el conjunto de datos de entrada contiene tanto datos etiquetados como no etiquetados. Su objetivo es el mismo que el del aprendizaje supervisado con la esperanza de mejorar el desempeño del entrenamiento al emplear los datos sin etiquetar [5].

## 1.3 Algoritmos de Agrupamiento

El agrupamiento, también conocido como clustering, es un proceso de machine learning que clasifica objetos en grupos o clusters basados en su similitud. A diferencia de otros métodos de aprendizaje supervisado, el agrupamiento no requiere etiquetas predefinidas; en cambio, utiliza algoritmos para descubrir estructuras internas en los datos. Este enfoque es particularmente valioso cuando se manejan conjuntos de datos grandes y complejos, donde la identificación de patrones y tendencias puede revelar información significativa. Existen disímiles formas de agrupar datos, siguiendo diferentes criterios, surgen numerosos algoritmos para cada uno[5].

### 1.3.1 Agrupamiento por distancias o centroides:

Los algoritmos de agrupamiento por distancias, como K-means, operan mediante la identificación de grupos o clusters de datos basados en la proximidad o distancia entre los puntos de datos. Estos algoritmos son fundamentales en el análisis de datos y la ciencia de datos, permitiendo la segmentación de conjuntos de datos en grupos coherentes sin necesidad de etiquetas previas [25].

Funcionamiento del Algoritmo K-means

El algoritmo K-means es uno de los métodos más populares de agrupamiento por distancias. Explicando su funcionamiento paso por paso:

1. Inicialización: Primero, se define el número de clusters deseados (denotado como "k"). Luego, se seleccionan k puntos aleatorios del conjunto de datos para servir como los centroides iniciales de los clusters.
2. Asignación de Puntos a Cluster: Cada punto de datos se asigna al cluster cuyo centroide (el promedio de los puntos en ese cluster) esté más cercano, utilizando una medida de distancia, como la distancia euclidiana.
3. Actualización de Centroides: Una vez que todos los puntos han sido asignados a un cluster, los centroides de cada cluster se actualizan calculando el nuevo centroide como el promedio de todos los puntos asignados a ese cluster.
4. Repetición del Proceso: Este proceso de asignación y actualización se repite hasta que los centroides ya no cambien significativamente entre iteraciones, lo que indica que los clusters han convergido a una configuración estable.

Ventajas: Es simple de entender e implementar, y suele ser rápido para conjuntos de datos grandes. También es útil para explorar la estructura de los datos y para la segmentación de clientes en marketing [25].

Desventajas: Requiere que el usuario especifique el número de clusters (k), lo cual puede ser difícil de determinar. Además, es sensible a la elección inicial de los centroides, lo que puede llevar a resultados diferentes dependiendo de la inicialización [25].



Figura Representación de K-means

### 1.3.2 Agrupamiento por Densidad

Los algoritmos de agrupamiento por densidad, como DBSCAN (Density-Based Spatial Clustering of Applications with Noise), se basan en la idea de que los clusters son regiones densas en el espacio de datos, separadas por regiones de menor densidad de puntos. A diferencia de los algoritmos de agrupamiento basados en distancias como K-means, DBSCAN no necesita que el número de clusters sea especificado previamente y puede manejar clusters de formas irregulares y de diferentes densidades [25,5,7].

**Funcionamiento del Algoritmo DBSCAN**

El algoritmo DBSCAN opera siguiendo estos pasos principales:

1. Selección de un Punto Inicial: Selecciona un punto arbitrario del conjunto de datos que aún no ha sido visitado.
2. Identificación de Vecinos: Determina los vecinos de este punto dentro de un radio definido por eps (epsilon), que es una medida de distancia.
3. Verificación de Densidad: Si el número de vecinos encontrados es mayor o igual a MinPts, se considera que el punto está en una región densa.
4. Expansión del Cluster: Si el punto seleccionado no está en ningún cluster, se añade a un nuevo cluster. Luego, el algoritmo expande este cluster incluyendo todos los puntos que están dentro de eps de cualquier punto en el cluster, siempre que tengan al menos MinPts.
5. Proceso Recursivo: Este proceso se repite para todos los puntos en el conjunto de datos hasta que todos hayan sido visitados.

Conceptos Clave

* Core Points: Son aquellos puntos que tienen al menos MinPts puntos dentro de eps.
* Border Points: Son puntos que están en la frontera de un cluster, teniendo al menos un punto Core a una distancia eps.
* Noise Points: Son puntos que no cumplen con las condiciones anteriores y, por lo tanto, no forman parte de ningún cluster.

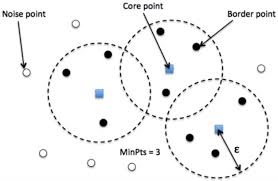


Figura Aplicación de DBSCAN y sus Conceptos claves

### 1.3.3 Agrupamiento por Jerarquía

Los algoritmos de agrupamiento jerárquico, como la Jerarquía Aglomerativa, se basan en la idea de construir una estructura jerárquica de clusters a partir de un conjunto de datos. Este proceso implica unir gradualmente los puntos más cercanos en clusters más grandes, creando una serie de niveles de agrupamiento. Los resultados típicos de este proceso se presentan en un dendrograma, que muestra la progresión de la unión de clusters a lo largo de diferentes niveles de agregación [25,5,7].

Funcionamiento de la Jerarquía Aglomerativa

La Jerarquía Aglomerativa sigue estos pasos clave:

1. Inicialización: Todos los puntos de datos individuales se consideran clusters en el nivel más alto de la jerarquía.
2. Cálculo de Distancias: Se calcula la distancia entre cada par de clusters. Esto puede hacerse usando medidas como la distancia euclidiana, la distancia de Manhattan, etc., dependiendo de la naturaleza de los datos.
3. Unión de Clusters: Los dos clusters más cercanos se unen para formar un nuevo cluster. Este proceso elimina los pares de clusters más cercanos y reduce el número total de clusters en uno.
4. Repeticiones: Este proceso de calcular distancias y unir clusters se repite hasta que solo quede un cluster grande que contiene todos los puntos de datos originales.
5. Visualización en Dendrograma: Los resultados finales se presentan en un dendrograma, que muestra la historia de las uniones de clusters a lo largo de los niveles de la jerarquía.

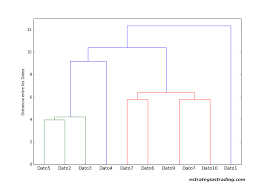


Figura Dendograma como método de representación las relaciones Jerárquicas

Ventajas y Desventajas

Ventajas: La Jerárquica Aglomerativa es útil para visualizar la estructura de los datos y para entender cómo los puntos de datos se relacionan entre sí. Es especialmente útil en situaciones donde la estructura de los clusters no es obvia o cuando se desea explorar múltiples niveles de agrupamiento [5].

Desventajas: Este método puede ser computacionalmente intensivo para conjuntos de datos grandes debido a su complejidad de O(n^3) para la versión general. Sin embargo, existen optimizaciones como SLINK y CLINK que reducen esta complejidad a O(n^2) para ciertos casos [5].

## 1.4 AutoML

La implementación de AutoML en tareas de aprendizaje no supervisado, como el agrupamiento, implica el uso de algoritmos que pueden seleccionar automáticamente los modelos más adecuados, ajustar sus hiperparámetros, realizar modelado iterativo y evaluar los modelos sin necesidad de una variable objetivo específica. Esto es especialmente útil cuando se trabaja con conjuntos de datos que no tienen etiquetas o categorías predefinidas para su análisis [22,16] .

En el contexto del agrupamiento, los algoritmos de AutoML pueden explorar un conjunto de datos basándose en diferentes características para agrupar ejemplos de manera que cada grupo contenga ejemplos lo más similares posible entre sí pero distintos de los ejemplos en otros grupos. La calidad de la separación resultante puede ser evaluada utilizando métricas de rendimiento, como el *Coefficent Silhouette*, que proporciona una medida cuantitativa de cómo están formados los clústeres y qué tan bien se diferencian entre sí [22,26].

Para implementar AutoML en tareas de agrupamiento, se puede seguir un enfoque modular donde se inicializa una clase con un conjunto de estimadores y transformadores. Este proceso incluye entrenar los estimadores y predecir la pertenencia a clusters para cada muestra. Las predicciones y métricas de rendimiento se almacenan para cada estimador utilizado, permitiendo una comparación efectiva entre ellos [22,26].

A pesar de la utilidad de AutoML para simplificar el proceso de modelado en tareas de agrupamiento, es importante tener en cuenta que la elección de los modelos y la configuración de los hiperparámetros pueden requerir un nivel de intervención manual dependiendo de la complejidad del problema y los requisitos específicos del proyecto. Además, aunque existen soluciones de AutoML que pueden automatizar parte del proceso de selección y ajuste de modelos, aún puede ser necesario un análisis detallado y una comprensión profunda de los datos y los objetivos del modelo para obtener resultados óptimos [26].

### 1.4.1 Pre-procesado

Uno de los primeros pasos dentro de cualquier proyecto de Machine Learning es la preparación de los datos, etapa que ocupa la mayor parte del tiempo, por lo que su automatización afecta de manera especial el factor tiempo, apoyando de igual manera al personal especializado y amateur [18]. Dentro del pre-procesado, se tiene en cuenta según [18]:

∙ Recolección de datos: la recolección de datos se puede tornar compleja según la tarea seleccionada. En muchos casos la insuficiencia de datos de entrada compromete de manera crítica la confiabilidad de los resultados. Internet ha facilitado esta tarea, sin embargo, generalmente se mantiene la dificultad para encontrar el material necesario, por lo que las herramientas de simulación de datos han tomado protagonismo. Esto se evidencia en diferentes campos como la industria automovilística, los videojuegos y medicina, apoyando la innovación en entornos emergentes de estas áreas [18].

∙ Limpieza de los datos: tradicionalmente, eliminar el ruido y obtener un conjunto de datos propicio para la ejecución de los diferentes algoritmos, requiere de la pericia de un profesional, dado el sin número de transformaciones necesarias. Los enfoques de AutoML en esta etapa se encuentran encaminados en dos líneas: primero, un proceso puntual de arreglar las bases de datos necesarias para el proyecto, y un segundo 14 enfoque de limpieza continua, a medida que se generan los datos, más afín con las necesidades empresariales [18].

La limpieza de los datos se realiza teniendo como objetivo el cumplimiento de los requisitos de los algoritmos a emplear, pero de manera general se recogen en la Tabla 1 las tareas y posibilidades de usar AutoML en las mismas:

|  |  |
| --- | --- |
| **Tarea de pre-procesado** | **Posibilidades de AutoML** |
| Manejo de datos de tipo fecha | Transformar todas las fechas a números, mediante la conversión de esta a milisegundos y luego el formateo de dicha columna a formato Integer. Esto permite el aprovechamiento de estos atributos y la distancia euclidiana |
| Manejo de datos de tipo String | Eliminación de las columnas string, ya que estos no funcionan con la distancia euclidiana |
| Manejo de datos tipo Integer, Long Double: | Normalización |
| Manejo de valores faltantes: | Eliminación de los valores faltantes en dependencia de cómo se manifieste |
| Dimensionality Reduction: El agrupamiento a menudo se aplica a conjuntos de datos de alta dimensionalidad. Técnicas como PCA (Análisis de Componentes Principales) pueden reducir la dimensionalidad manteniendo la varianza máxima de los datos.  Clustering Previo: Antes de aplicar un algoritmo de agrupamiento, a veces se utiliza un paso previo de clustering para identificar y eliminar valores atípicos o outliers, lo que puede mejorar la calidad de los clusters generados posteriormente. | Reducción de Datos (Data Reduction) |
| Identificar y tratar valores atípicos o outliers es crucial antes de aplicar algoritmos de agrupamiento. Los outliers pueden distorsionar los clusters y afectar negativamente el rendimiento del modelo. Técnicas como el clustering parcial pueden ser útiles para detectar y manejar estos valores. | Detección de Datos Anómalos |

Tabla Técnicas de Pre-procesamiento en tareas de agrupamiento

### 1.4.2 Procesado (Optimización de Hiperparametros)

Cada sistema de Machine Learning tiene varios hiperparámetros que especificar, cuya elección tiene un gran impacto en el rendimiento. Los algoritmos están ligados a un conjunto de hiperparámetros, ya sean números enteros, binarios, de valor real o, en algunos casos, condicionales, que pueden afectar en gran medida resultados como el número de capas ocultas en una red neuronal profunda [18,26].

En consecuencia, optimizar el conjunto de valores de los parámetros de entrada del algoritmo es la tarea principal de cualquier sistema AutoML y está vinculado al problema de selección del algoritmo en ese contexto. Se aplican varios casos de uso a HPO, ya que reduce el esfuerzo humano necesario para aplicar el aprendizaje automático, mejora el rendimiento de los algoritmos de aprendizaje automático y mejora la reproducibilidad y la equidad de los estudios científicos [26].

Por lo que cada aplicación de la HPO(*Hyperparameter Optimization*) varía en dependencia de los parámetros de entrada de cada algoritmo a utilizar. En el capítulo 2 se muestra la aplicación por separado en cada caso [26].

### 1.4.3 Representación de Resultados

La representación de resultados en los flujos de Clustering, un área del Machine Learning, es crucial para entender cómo los datos han sido agrupados y qué significa cada agrupamiento en términos prácticos. Este proceso implica visualizar y analizar los clusters generados por los algoritmos de clustering para identificar patrones, tendencias y relaciones entre los datos [22].

A continuacion algunos aspectos claves a considerar:

**Visualización de Clusters**

La visualización de clusters es uno de los métodos más efectivos para representar los resultados de un proceso de clustering. Dependiendo de la dimensionalidad de los datos, esto puede realizarse mediante gráficos bidimensionales (para datos de baja dimensión) o mediante técnicas de visualización avanzadas como mapas de calor, gráficos de dispersión, o incluso visualizaciones tridimensionales para datos de mayor complejidad [22].

* Gráficos Bidimensionales: Ideal para datasets con dos características. Los puntos se colorean o se etiquetan según el cluster al que pertenecen, permitiendo una rápida identificación de grupos.
* Mapas de Calor: Útiles para visualizar la densidad de los datos en relación con los clusters, mostrando áreas de alta concentración de puntos de datos.

**Análisis de Resultados**

El análisis de los resultados del clustering va más allá de la simple visualización. Implica interpretar qué significan los clusters en el contexto del problema que se está abordando. Esto puede incluir:

* Identificación de Patrones: Buscar patrones comunes dentro de cada cluster que puedan indicar características compartidas entre los elementos del mismo grupo.
* Relación entre Clusters: Analizar la similitud y diferencia entre clusters para entender cómo se relacionan entre sí y qué factores podrían influir en estas relaciones.
* Validación de Clusters: Utilizar métricas específicas de clustering, como la silueta, para evaluar la calidad de los clusters formados. Una silueta bien definida indica que los elementos dentro de un cluster son similares entre sí y diferentes de aquellos en otros clusters.

**Consideraciones Importantes**

* Elección de Métricas de Proximidad: La elección de la métrica de proximidad (por ejemplo, distancia euclidiana, distancia de Manhattan) es crucial, ya que afecta cómo se calcula la similitud entre los puntos de datos y, por ende, cómo se forman los clusters.
* Número de Clusters: Determinar el número óptimo de clusters es un desafío común en el clustering. Técnicas como el método del codo o la función de calinski-harabasz pueden ayudar a encontrar este número.
* Tipo de Datos: La naturaleza de los datos (continuos, discretos, binarios) influye en la elección del algoritmo de clustering y en la forma en que se deben representar los resultados.

## 1.5 KNIME

KNIME es una herramienta de Ciencia de Datos, de código abierto. En su página web se define como: “En KNIME, creamos software para crear y producir ciencia de datos utilizando un entorno fácil e intuitivo, lo que permite que todas las partes interesadas en el proceso de ciencia de datos se centren en lo que hacen mejor” [24].

KNIME se encuentra disponible con versiones para Windows, Linux y Mac OSX. En el caso de Windows se encuentra en un entorno de Eclipse, con Java como lenguaje principal. La amplia variedad de nodos disponibles permite completar casi cualquier tarea sin programación adicional; pero, de ser necesario existen nodos que habilitan la programación en Java [24, 25]. Igualmente presenta, ya sea de manera nativa o extendida de otras herramientas, un gran número de algoritmos de Minería de Datos [26].

KNIME basa su funcionamiento en los nodos y variables de flujos. La interconexión de estos dos elementos genera un flujo, que a su vez puede ser encapsulado en metanodos y componentes.

Los nodos representan una tarea en específico. KNIME agrupa nodos que ejecutan diferentes tareas como lectura/escritura de archivos, transformación de datos, entrenamiento de modelos, visualización (tablas y gráficos), etc. En dependencia del nodo puede presentar uno, varios o ningún puerto de entrada y salida [18]. Presenta la particularidad de mantener su configuración al ser copiado.

Los flujos son secuencia de nodos que realizan una tarea. Generalmente comienzan como un nodo que lee los datos a analizar [18]. Una característica de KNIME es que cada nodo mantiene el estado resultante de su ejecución, por lo que es posible guardar cada estado de un flujo como se evidencia en la Figura 8. Adicionalmente, los flujos se ejecutan secuencialmente y cada nodo emplea únicamente los datos proporcionados por su conexión directa [24].

Este trabajo pretende crear varios componentes que sean capaces de acoplarse en una mismo funcionando con un orden específico y este sea capaz de integrarse a otros flujos de Knime.

En KNIME, un componente es una funcionalidad que encapsula tareas reutilizables que pueden ser utilizadas como nodos personalizados de KNIME. Estos componentes pueden ser creados por usuarios para realizar tareas que se repiten frecuentemente o simplemente almacenados para su uso futuro. Los componentes pueden compartirse con otros a través del KNIME Hub y el KNIME Server, facilitando la colaboración y el intercambio de soluciones analíticas predefinidas [24].

**Los componentes ofrecen varias ventajas sobre los nodos nativos de KNIME:**

* Reusabilidad: Permiten encapsular lógica compleja en un único nodo, haciendo que los flujos de trabajo sean más fáciles de entender y mantener.
* Personalización: Pueden ser configurados para adaptarse a diferentes casos de uso, ajustando parámetros internos sin modificar el código subyacente.
* Compartición: Facilitan la colaboración al permitir compartir componentes completos, incluyendo su configuración y dependencias, a través de plataformas como KNIME Hub y KNIME Server.

Además, los componentes pueden incluir nodos de widget dentro de ellos para construir vistas compuestas que se visualizarán como páginas web en el KNIME WebPortal. Esto permite establecer configuraciones específicas antes de ejecutar el flujo de trabajo, inyectando valores específicos en el flujo de trabajo para parametrizar su ejecución.

##### 

Figura Ejemplo de flujo KNIME

## 

## Conclusiones parciales

AutoML provee grandes ventajas en el desarrollo de cada etapa en un proyecto de Machine Learning, proporcionando soporte en la toma de decisiones desde el pre-procesado hasta la selección de hiperparámetros y visualización de los resultados. La minería de datos es uno de los campos que se benefician de las bondades de AutoML; una herramienta que presenta gran afinidad con ambas áreas es KNIME. El agrupamiento es una de las tareas de Minería de Datos más empleadas en la actualidad; mientras que K-means, DBSCAN y Hierarchical Clustering son de los modelos más empleados en el desarrollo de esta tarea, ya que representan las principales metodologías de agrupamiento. A pesar de sus diferencias y mejoras, los 3 poseen rendimientos similares, por lo que su uso queda determinado por el conjunto de datos a emplear.

# Capítulo 2: Desarrollo del Componente AutoML Clustering

En este capítulo se especifican los detalles que componen el diseño del Componente General: “AutoML Clustering” y los subcomponentes “K-Means HPO”,”DBSCAN HPO”,”Hierarchical HPO”.

Al igual que los mecanismos de pre-procesado y representación de los resultados.

## 

## 2.1 Mecanismo de pre-procesado

Con el fin de abordar las técnicas identificadas en el capítulo anterior. Las cuales recogen las tareas: Recolección de datos, Limpieza de los datos, Manejo de los tipos de datos(fechas), Manejo de valores faltantes. Se aprovechan en cada caso las capacidades de Knime para construir un bloque inicial de nodos encargados de hacer llegar los datos a los subcomponentes de procesamiento, la información con la calidad necesaria para evitar errores de compilación y conseguir un resultado de calidad tanto en métricas como en la representación de este.

### 2.1.1 Recolección de datos

Para mejorar el proceso de AutoML, en la etapa del pre-procesado también se emplea el enfoque de la recolección de parámetro introducidos por el usuario, los cuales surtirán efecto a lo largo de él flujo. Estos parámetros son:

* Graficar Clusters: Graficar los datos con respecto a los clusters generados en el Eje Y, con respecto a otro parámetro elegido para el eje X.
* Eje Y: En el caso de no elegir “Graficar Clusters”, el usuario elige qué atributo del dataset utilizar para el Eje Y
* Eje X: El usuario elige el atributo del dataset a utilizar para graficar para el Eje X.
* Selección de Modelos: El usuario elige los modelos a entrenar(K-Means, DBSCAN, Hierarchical Clustering)

Empleando los nodos Boolean Configuration, Column Selection Configuration y Multiple Selection, entre otros(figura 5). Knime permite que el usuario que tenga el relativo conocimiento de la tarea a realizar reduzca el tiempo de compilación en tareas innecesarias (ejemplo: eligiendo solo el o los modelos que necesita entrenar). También esta estrategia permite el control de la fase de representación de resultados desde el preprocesamiento del componente

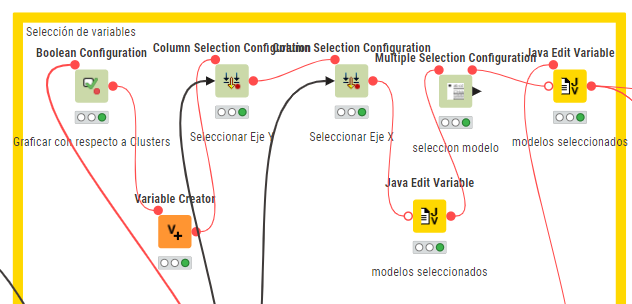


Figura Recolección de Datos

### 2.1.2 Pre-procesado de datos tipo string

Los valores nominales únicos, o categorías con un solo ejemplo en una columna, pueden parecer insignificantes a primera vista, pero su correcto manejo es esencial para evitar posibles problemas de calidad de datos y garantizar la integridad de los análisis. El componente: “*AutoML Clasificación (pre-procesado)”* (tomado como referencia) poseía un tratamiento erróneo de estos valores al eliminar las columnas nominales que contenían un número de categorías diferentes que superan un umbral. Este umbral era determinado por el usuario en la configuración del componente [24,26].

En aras de depurar estas inconsistencias, el diagrama de la Figura 2.3 muestra un nuevo flujo para el pre-procesado de *string*, donde la actividad de color amarillo expresa que se modificó la que anteriormente se encontraba en el componente; mientras que la actividad en verde refleja una nueva implementación.



Figura 2.3: Diagrama de flujo para el pre-procesado de *string*

A continuación se exponen los cambios realizados al componente *String preprocs*:

Eliminar valores únicos por columna: se enmienda el error antes expuesto, al modificar el componente *Filtrar valores únicos*, implementando la eliminación de las columnas donde más del 80 % de los valores son únicos. En la Figura 6 se muestra una parte del flujo KNIME donde se implementa esta tarea [24,26].

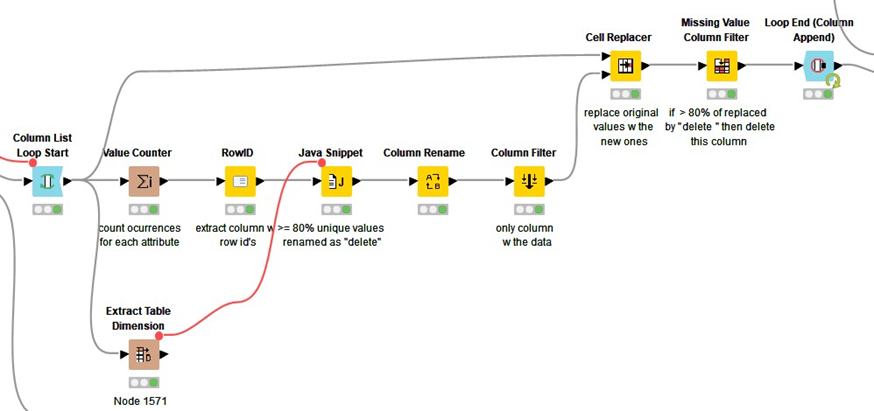


Figura Vista previa de flujo KNIME para el filtrado de valores únicos

Reemplazar con ’otros’: en este nuevo componente, los valores únicos en una columna que representan una minoría, son reemplazados por la categoría ’*other*’. Para ello, iterando por cada columna dentro de un ciclo, se calculan la frecuencia absoluta y relativa de cada atributo (nodo *Math Formula*), y aquellos que representen menos del 1 % son los elegidos para la sustitución por la nueva categoría (nodo *Java Snippet*). En la Figura 7 se muestra una parte del flujo KNIME donde se implementa esta tarea [24,26].

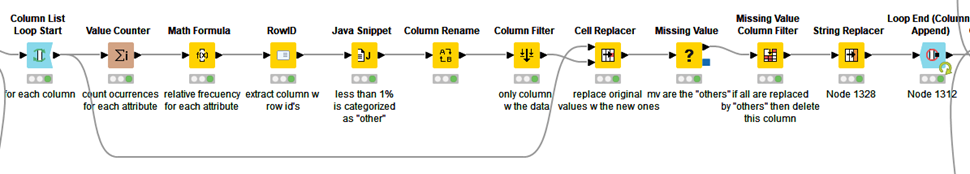


Figura Vista previa de flujo KNIME para el reemplazo por ’other’

### 2.1.3 Manejo de valores faltantes

El tratamiento de valores faltantes en conjuntos de datos es un paso crítico en la preparación y análisis de datos. La presencia de datos faltantes puede afectar significativamente la calidad y la fiabilidad de cualquier análisis o modelo que se derive de ellos. Adicionalmente, algunos algoritmos requieren que no existan valores faltantes para su funcionamiento. En aras de mejorar la imputación de estos valores, se propone modificar el componente *Valores faltantes*, cuyo diagrama de flujo se presenta en la Figura 8 [24,26].



La imputación de valores faltantes ha sido modificada, dado que anteriormente se sustituyen por la media los valores faltantes numéricos, y por la moda los atributos categóricos. En la Figura 2.7 se presenta el diagrama de flujo de la nueva implementación para la sustitución de estos valores [24,26].

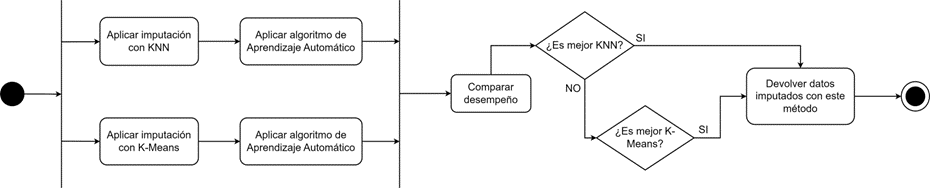


Figura Diagrama de flujo para la imputación de valores faltantes

Primeramente se aplican en paralelo las técnicas de imputación kNN y k-Means, al terminar se aplica el algoritmo de aprendizaje automático en cuestión con los datos imputados por cada método y, finalmente, se comparan los resultados acorde al esquema descrito en el componente *Discretizer*, para devolver los datos con la mejor sustitución [24,26].

Para la implementación de kNN y k-Means, se crearon dos subcomponentes: kNNI (*k- Nearest Neighbors Imputation*) y kMI (*k-Means Imputation*). Se emplean los nodos *kNN* y *k-Means*, ambos nativos de KNIME. Para escoger *k*, en k-Means se utiliza el Método de la Silueta, ya que KNIME contiene el nodo *Silhouette Coefficient*; mientras que para kNN se decide emplear la raíz cuadrada de la muestra, debido a su simpleza en la implementación y que este método podría ofrecer cierta estabilidad en la elección del número de vecinos, independientemente del tamaño específico del conjunto de datos. Esto podría hacer que el modelo sea menos sensible a variaciones en el tamaño de la muestra. A continuación se describe el funcionamiento de kNNI y kMI [24,26]:

**kMI:**

1. Convertir los valores nominales en numéricos para que el nodo trabaje con ellos, ya que este algoritmo solamente trata este tipo de valores.

2. Separar las columnas con valores perdidos para que el algoritmo pueda realizar el proceso de *clustering* con los datos sin estos valores, ya que no los tolera el nodo.

3. Calcular valor óptimo de *k* con el Método de la Silueta.

4. Aplicar k-Means a estos datos.

5. Unir las columnas que contienen valores perdidos con los datos etiquetados con su respectivo clúster.

6. Aplicar el nodo *Missing Value*, nativo de KNIME, tras filtrar por clúster, sustitu- yendo los valores perdidos por la media de esa columna, es decir el valor del centroide de ese clúster.

7. Retornar las variables numéricas a categóricas (las que se transformaron inicial- mente) para recuperar su valor original.

**kNNI:**

1. Convertir los valores nominales en numéricos para que el nodo trabaje con ellos, ya que este algoritmo solamente trata este tipo de valores.

2. Extraer los nombres de las columnas que contienen valores perdidos.

3. Por cada columna, se separan los valores perdidos del resto de los datos, éstos se reemplazan por 0 si son numéricos, si son de tipo *string* se ignoran.

4. Calcular el valor de *k* mediante la raíz cuadrada de la muestra.

5. Los datos sin valores perdidos se emplean para el entrenamiento de kNN, mien- tras los datos con estos se emplean para la predicción.

6. Retornar las variables numéricas a categóricas (las que se transformaron inicial- mente) para recuperar su valor original.

En las Figuras 9 y 10 se muestran los flujos de trabajo que se implementan para la imputación con los algoritmos kNNI y kMI, respectivamente.

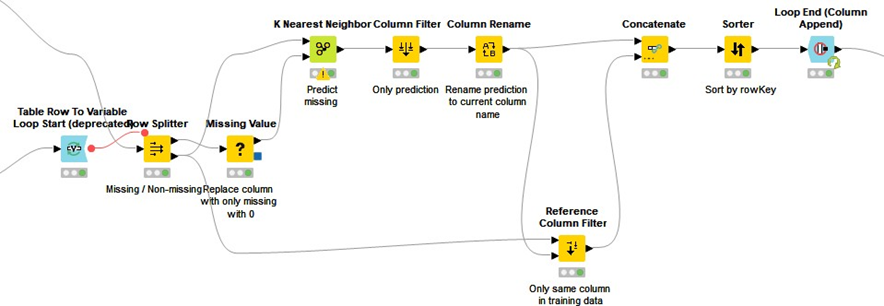


Figura Vista previa de flujo KNIME para kNNI

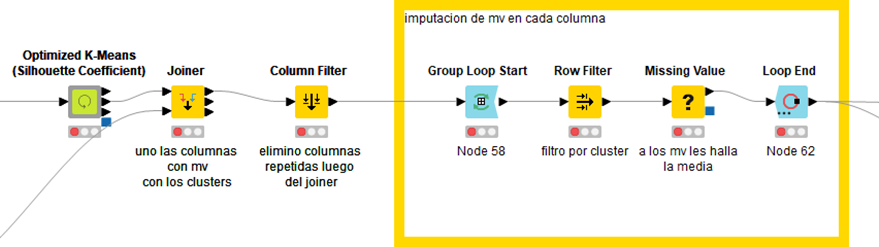


Figura Vista previa de flujo KNIME para kMI

### 2.1.4 Codificación y Normalización

La codificación de variables categóricas implica asignar valores numéricos a estas cate- gorías para que los algoritmos de aprendizaje automático puedan trabajar con ellas. Este es el propósito del componente *Pre-procesar números*, además de la normalización de variables numéricas. En aras de clarificar el funcionamiento de este proceso, se decide renombrarlo a *Codificar y normalizar* [24,26].

La normalización son proporciones sin unidades de medida (adimensionales o invariantes de escala) que nos permiten poder comparar elementos de distintas variables y unidades de medida. Esta es necesaria para cambiar los valores de las columnas numéricas del conjunto de datos para usar una escala común, sin distorsionar las diferencias en los intervalos de valores ni perder información. La normalización es fundamental para que algunos algoritmos modelen los datos correctamente [24,26].

En KNIME es posible implementar la normalización a partir del nodo *Normalizer*. En este nodo se encuentran tres métodos para normalizar, a elección del usuario: *Decimal Scaling, Z-Score y Min-Max*. Este proceso se encuentra en el componente para el pre-procesado de números, cuyo diagrama de flujo se presenta en la Figura 11.

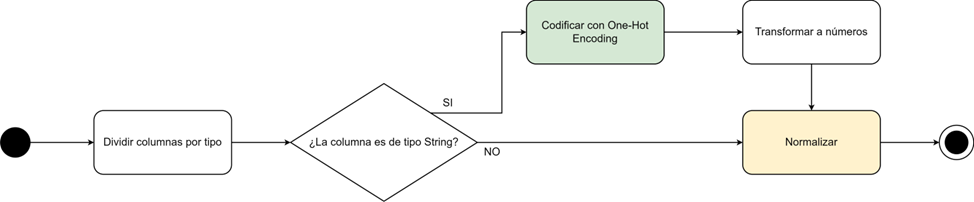


Figura Proceso de Normalización

pueden ser numerosos, son cruciales para comprender tendencias y patrones en los datos. La codificación de estas variables se refiere a técnicas especiales de codificación que se utilizan cuando se tienen variables categóricas con un gran número de categorías o niveles distintos. La alta cardinalidad puede dificultar la gestión de estas variables en modelos de aprendizaje automático, y es importante abordarlas de manera eficiente.

A continuación, se exponen los cambios realizados al pre-procesado para la codificación y normalización:

* Codificar con *One-Hot Encoding*: para el tratamiento de valores con alta cardinalidad, se emplea la codificación One-Hot, ya que KNIME brinda el nodo *One To Many* para esta tarea. Para esto, se escogen los atributos que tienen como mínimo 15 categorías distintas. *One-Hot Encoding* se utiliza para codificar variables categóricas en una forma que no impone un orden implícito en las categorías. Cuan- do se tienen más de 15 categorías, es poco probable que haya un orden natural o jerarquía en esas categorías, por lo que codificarlas como variables binarias evita in- terpretaciones erróneas de orden o importancia. Dado que esta técnica produce gran dimensionalidad, se utiliza para su reducción el método PCA, con un límite de con- servación de la información del 90 %. Este proceso se encuentra implementado en la Figura 12.

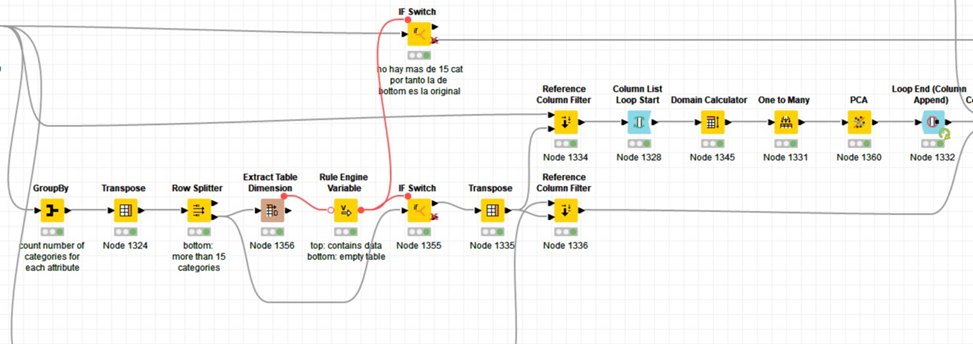


Figura Vista previa de flujo KNIME para la codificación One-Hot

* Normalizar: con el propósito de automatizar este proceso, se propone el componen- te *Normalizer*, de igual nombre al nodo nativo de KNIME, en donde se encuentra el mismo para la ejecución de los métodos que contiene, en función de un modelo pre- determinado. En la Figura 13 se presenta el diagrama de flujo de este componente.

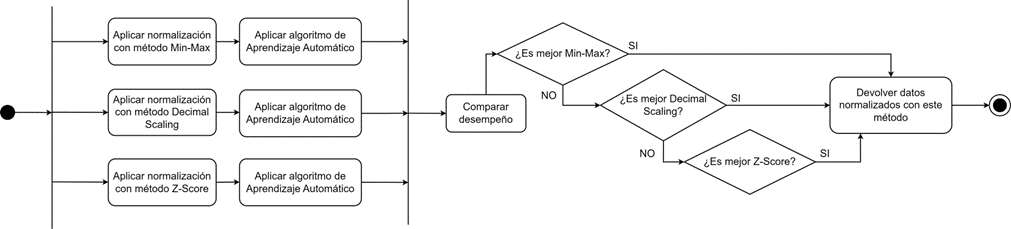


Figura Diagrama de flujo para la normalización

Siguiendo el mismo esquema del componente *Discretizer*, tras aplicar la normalización con los distintos métodos ofrecidos por la herramienta KNIME, se aplica el algoritmo de aprendizaje automático que requiere los datos normalizados y posteriormente, se realiza la evaluación del desempeño de cada uno, acorde a las métricas Exactitud y Cohen’s Kappa. Luego de escoger el mejor, se devuelve la tabla con los datos normalizados.

## 2.2 Prototipo del Componente AutoML Clustering

Antes de explorar en detalle las adaptaciones realizadas en cada modelo, es fundamental presentar el componente general para la optimización de hiperparámetros, con el objetivo de facilitar el entendimiento en los epígrafes posteriores. Los modelos a optimizar en el componente propuesto en este acápite son: K-Means, DBSCAN, Hierarchical Clustering(Aglomerativo). La selección adecuada de hiperparámetros desempeña un papel fundamental en el desarrollo de modelos de aprendizaje automático y estadísticos. Los hiperparámetros que se incluyen en un modelo no solo afectan su capacidad para comprender patrones y tomar decisiones precisas, sino que también pueden influir significativamente en la eficiencia computacional y los recursos requeridos. Al elegirlos correctamente, se simplifica y mejora la interpretación de los modelos, reduce el riesgo de sobreajuste y acelera el tiempo de entrenamiento. Algunos hiperparámetros que influyen en el rendimiento de los modelos son (Montavon et al., 2012), (Scholkopf & Smola, 2018), (Lakshmanan et al., 2021), (Hastie et al., 2009), (Gupta et al., 2017):

* K-Means: Cantidad de Clusters, columnas a utilizar
* DBSCAN: Valor de Epsilon, columnas a utilizar, cantidad mínima de puntos
* Hierarchical: Número de salida de clusters, función distancia, tipo de link, columnas a utilizar

##### **Configuración del modelo**

Ajustar las configuraciones de un modelo es clave para lograr un rendimiento óptimo, lo que a su vez mejora la precisión y la eficacia de sus análisis. En este contexto, se propone el componente *AutoML Clustering* presente en la Figura 14, conteniendo la siguiente configuración:

## 

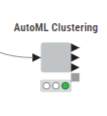


Figura Componente AutoML Clustering

1.Puerto de entrada: recibe los datos de entrada en formato tabular.

2. Elementos de la configuración:

* Graficar Clusters: Graficar los datos con respecto a los clusters generados en el Eje Y, con respecto a otro parámetro elegido para el eje X.
* Eje Y: En el caso de no elegir “Graficar Clusters”, el usuario elige qué atributo del dataset utilizar para el Eje Y
* Eje X: El usuario elige el atributo del dataset a utilizar para graficar para el Eje X.
* Selección de Modelos: El usuario elige los modelos a entrenar(K-Means, DBSCAN, Hierarchical Clustering)

3. Puertos de salida:

* Puerto 1: Tabla con los datos agrupados por K-Means, cada fila con su respectiva asignación de cluster.
* Puerto 2: Tabla con los datos agrupados por DBSCAN, cada fila con su respectiva asignación de cluster.
* Puerto 3 Tabla con los datos agrupados por Hierarchical Clustering, cada fila con su respectiva asignación de cluster.
* Mejor Modelo según Silhouette Coefficient

### 2.2.1 Requisitos del Componente AutoML Clustering

El componente propuesto debe cumplir los siguientes requisitos funcionales (RF):

RF1: el componente debe permitir seleccionar estrategia de optimización de hiperpa- rámetros.

RF2: el componente debe permitir seleccionar la columna a utilizar para la graficacion(Eje X, Eje Y)

RF3: el componente debe permitir seleccionar uno o varios de los algoritmos listados.

RF4: el componente debe entrenar y optimizar hiperparámetros para K-Means.

RF5: el componente debe entrenar y optimizar hiperparámetros para DBSCAN.

RF6: el componente debe entrenar y optimizar hiperparámetros para Hierarchical Clustering.

RF7: el componente debe graficar los resultados de los modelos seleccionados.

RF8: el componente debe retornar el mejor modelo de los entrenados.

El componente propuesto presenta las siguientes restricciones para su funcionamiento: Los datos de entrada deben encontrarse en formato tabular.

El Dataset debe tener al menos 1 columna de tipo numérico

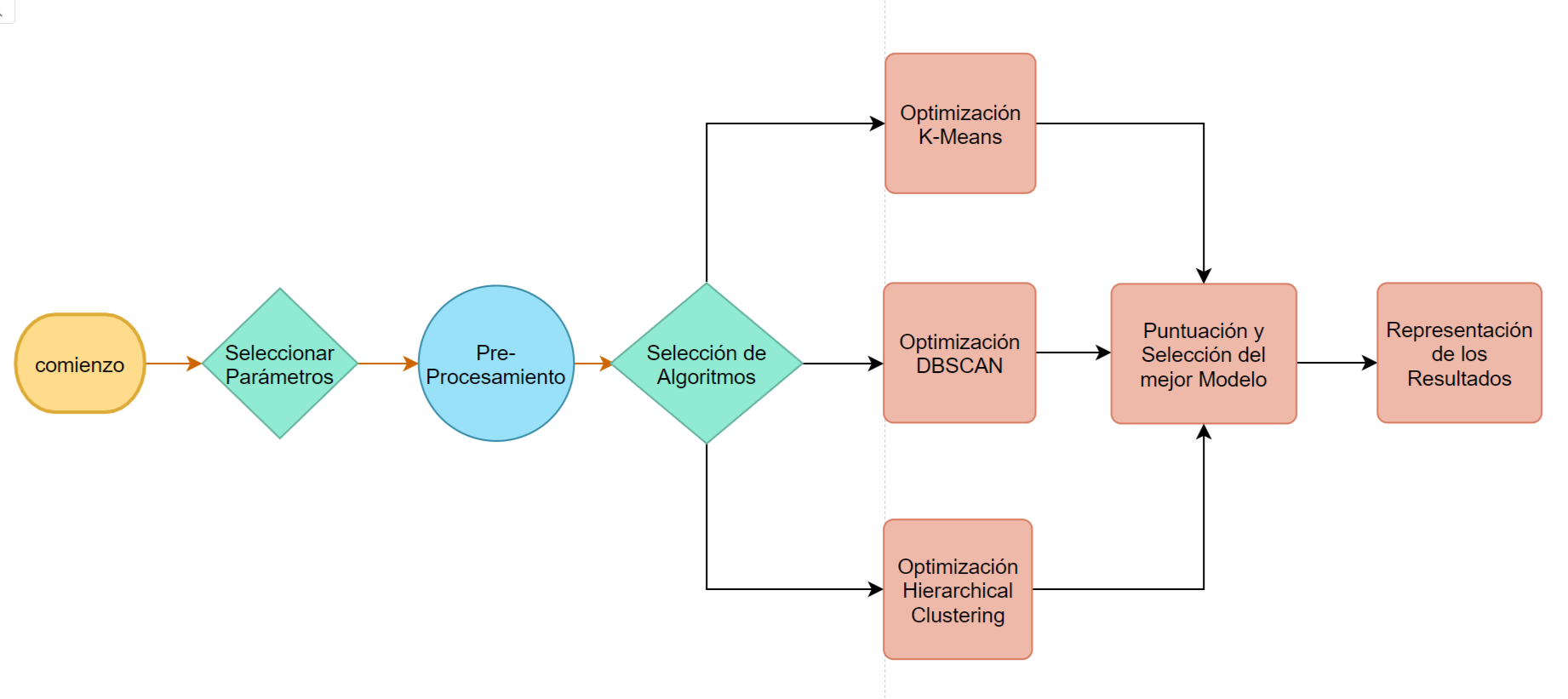


Figura Diagrama de flujo general del componente AutoML Clustering

##### ***Optimización de hiperparámetros***

El diagrama de flujo de la Figura 16 expone el flujo general del apartado de optimi- zación del componente *AutoML Clustering*

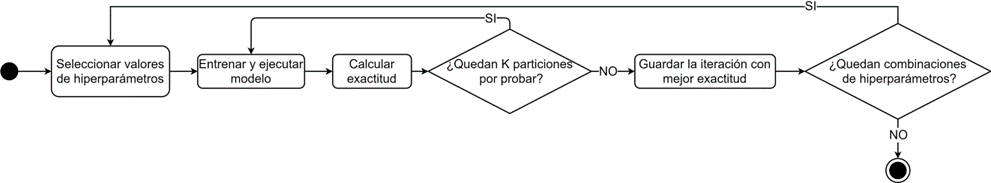
**

Figura Diagrama de flujo de la optimización de hiperparámetros

En la Figura 2.16, la optimización de hiperparámetros implica la reiteración sistemática a través de un conjunto predefinido de hiperparámetros, evaluando exhaustivamente todas las combinaciones posibles. Este proceso se detiene una vez que se han evaluado todas las combinaciones. Cada iteración se somete a una validación cruzada, que implica la evaluación de los hiperparámetros en múltiples subconjuntos de prueba. La métrica de rendimiento (en este caso, la exactitud) se calcula para cada iteración de la validación cruzada, y se registra la iteración con la mayor exactitud. Finalmente, una vez que se han evaluado todas las combinaciones posibles, se devuelve la iteración del modelo que obtuvo la mejor exactitud [24,26].

En KNIME es posible implementar el ciclo para comprobar las combinaciones de hiper parámetros mediante los nodos Parameter Optimization Loop Start y Parameter Optimization Loop End, los cuales permiten almacenar las iteraciones realizadas por el algoritmo con las diferentes combinaciones de hiperparámetros dentro de un rango previamente establecido [24,26].

## 2.3 Optimización y Procesamiento de K-Means

Para el entrenamiento y prueba de K-Means, se emplea el componente de la comunidad Optimized K-Means (Silhouette Coefficient), el cual emplea las configuraciones siguientes, en torno a optimizar el hiperparámetro del valor k:

* **Valor Inicial para K**: Valor inicial para iniciar las iteraciones y pruebas para encontrar el valor óptimo de k(número de clusters con los que agrupar).
* **Máximo número de iteraciones:** Maximo numero de iteracion partiendo del valor inicial de k, utilizando el valor de cantidad de pasos por iteración.
* **Valor de pasos para k**: Cantidad de pasos por cada iteración en valor de k.
* **Selección de columnas**: Seleccionar las columnas en base a las cuales se agrupara.
* **Semilla de inicialización para la estrategia de búsqueda:** valor inicializador para la estrategia de búsqueda.
* **Estrategia de Optimización:** Fuerza Bruta o Escalador de Colinas.

El diagrama de actividades de la Figura 17 y 18, expone el flujo para el procesamiento necesario para la ejecución del algoritmo K-Means, con optimización de hiperparámetros.

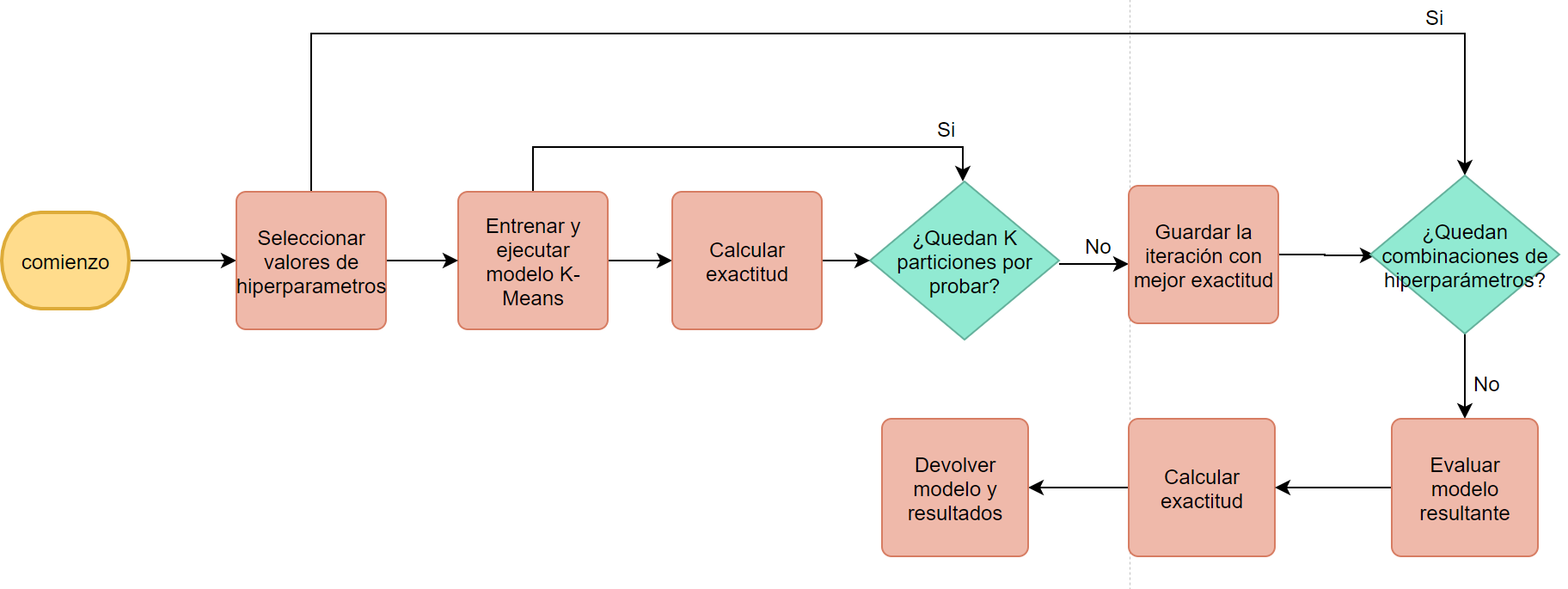


Figura Diagrama de flujo para la optimización de hiperparámetros de K-Means

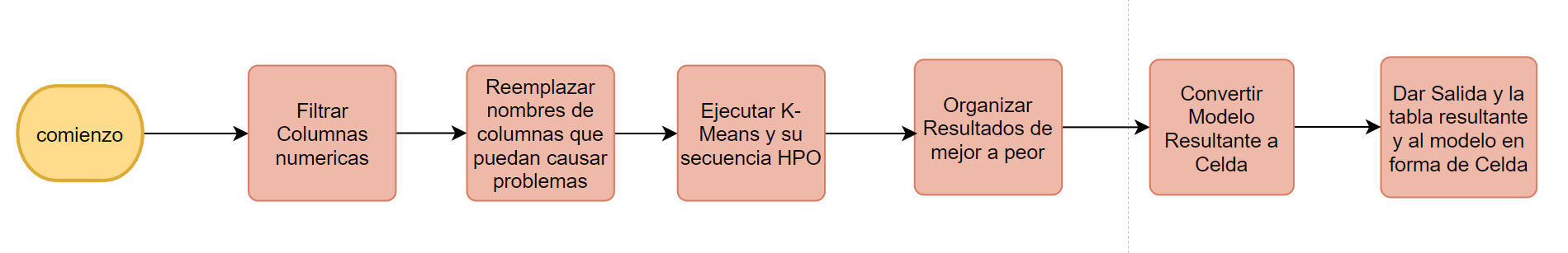


Figura Diagrama de flujo de procesamiento de K-Means

## 2.4 Optimización y Procesamiento de DBSCAN

Para el entrenamiento y prueba de DBSCAN, se emplea el componente de la librería Weka DBSCAN, el cual emplea las configuraciones siguientes:

* **epsilon:** Este valor genera el radio que tendrán cada uno de los puntos a usar para encontrar los puntos que tengan la cantidad mínima de vecino(*minPoints*)
* **minPoints:** Cantidad de vecinos mínimos que debe tener cada punto para considerarse cluster, dentro de su radio(*epsilon*).

El diagrama de actividades de la Figura 19, expone el flujo para el procesamiento necesario para la ejecución del algoritmo DBSCAN, con optimización de hiperparámetros. La estrategia tiene como enfoque encontrar el mejor valor de cada hiperparametros a base ejecucion y prueba, primero entrando con una combinación de valores minima, el valor óptimo de epsilon, luego pasa a la búsqueda del minPoints óptimo, en ambos casos pasando de un valor mínimo hasta un valor máximo razonable [24,26].



Figura Diagrama de flujo para la optimización de hiperparámetros de DBSCAN

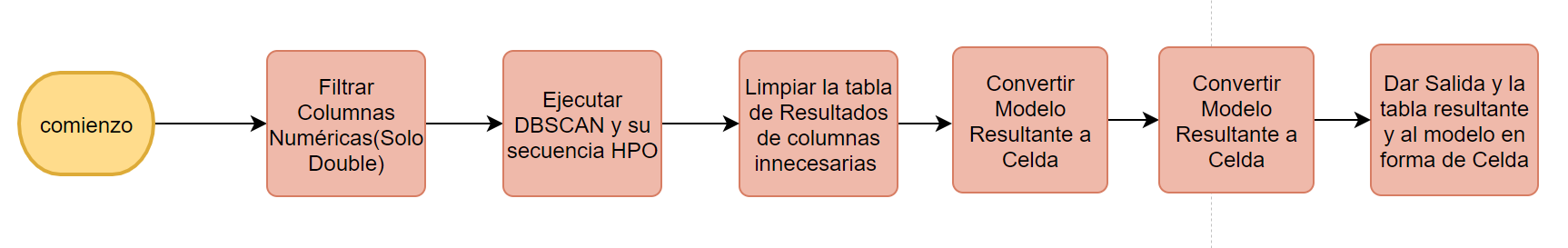


Figura Diagrama de flujo de procesamiento de DBSCAN

## 2.5 Optimización y Procesamiento de Hierarchical

Para el entrenamiento y prueba de Hierarchical, se emplea el componente de la librería Weka Hierarchical Clustering( solamente funciona de forma aglomerativa), el cual emplea las configuraciones siguientes:

* **Distance function:** Este parámetro se refiere a el tipo de distancia que utiliza el algoritmo para agrupar los datos.
* **link type:** Este parámetro se refiere al criterio utilizado para determinar cuánto se deben fusionar dos grupos al momento de realizar la agrupación.
* **número de clusters:** Al igual que en el K-Means, este parámetro representa el valor k, para este caso en particular el número final de clusters a él que debe llegar el algoritmo, comenzando de más a menos.

El diagrama de actividades de la Figura 21 y 22, expone el flujo para el procesamiento necesario para la ejecución del algoritmo Hierarchical, con optimización de hiperparámetros. La estrategia tiene como enfoque encontrar el mejor valor de cada hiperparametros a base ejecución y prueba, primero entrando con una combinación de valores minima, el valor óptimo de epsilon, luego pasa a la búsqueda del minPoints óptimo, en ambos casos pasando de un valor mínimo hasta un valor máximo razonable [24,26].

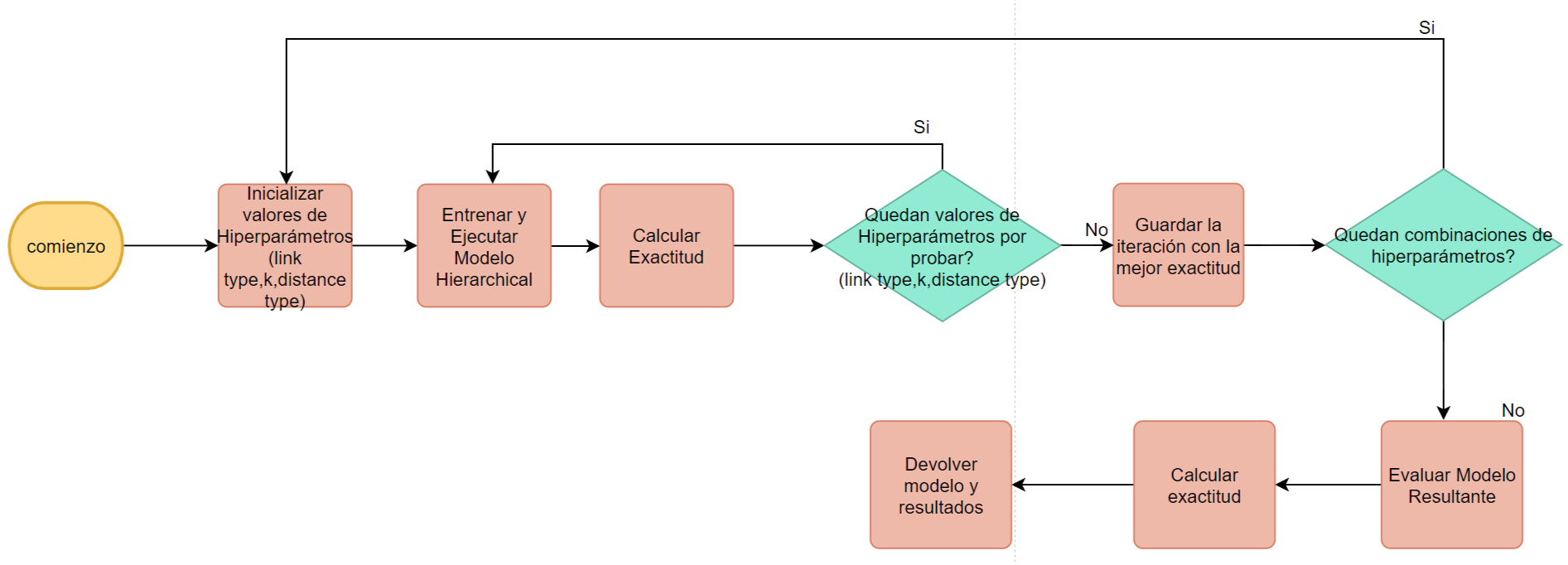


Figura Diagrama de flujo para la optimización de hiperparámetros de Hierarchical

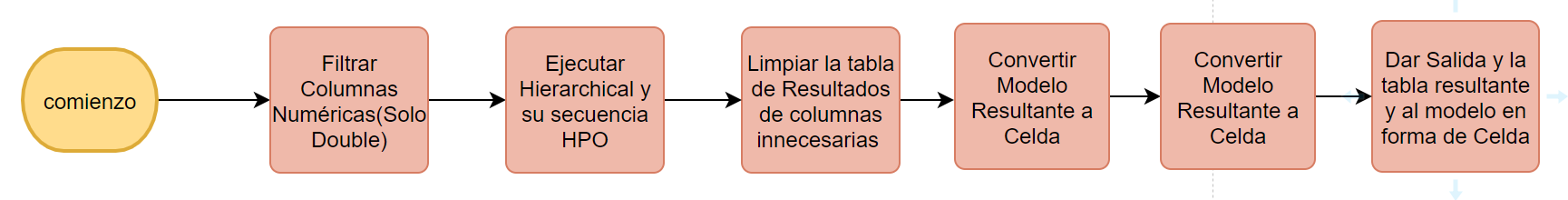


Figura Diagrama de flujo para el procesamiento de Hierarchical

## 2.6 Post-Procesamiento, puntuación y representación de los resultados

Luego del procesamiento de los datos por los diferentes algoritmos, el componente continua con la fase de post-procesado con los resultados. En esta fase se puntúan y representan los resultados de cada modelo, también para encontrar el mejor y devolverlo en los puertos de salida del componente.

### 2.6.1 Puntuación de los resultados(Componente Scorer)

Para puntuar los resultados y compararlos, se construye el componente *Scorer*, basando su funcionamiento en los resultados del Coefficient Silhouette utilizado de forma individual con cada modelo generado, en la figura 23 podemos ver la secuencia de selección del modelo salida basándose en los resultados de esta métrica y luego la graficación usando estos datos para representar la efectividad de cada algoritmo para el dataset utilizado.

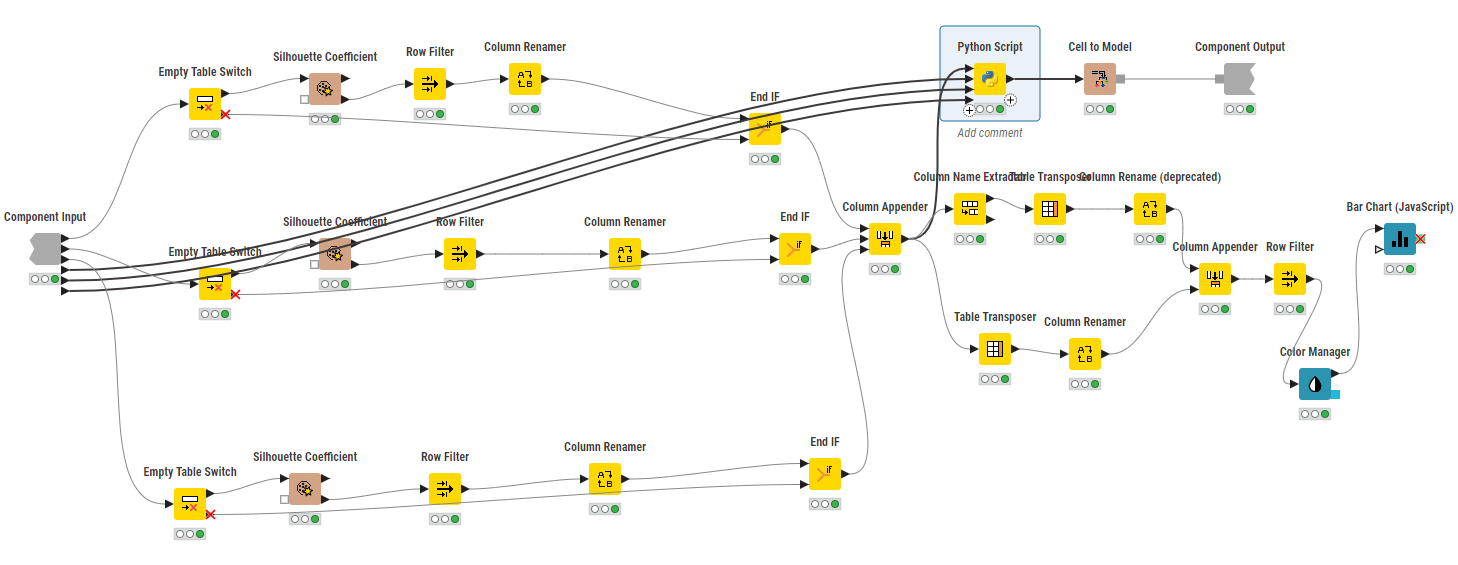


Figura Funcionamiento de Scorer

Cabe destacar el nodo resaltado(Python Script), el cual contiene una porción de código en Python, con el objetivo de seleccionar el mejor modelo de una forma más eficiente y consumiendo menos recursos. Además de el nodo Bar Chart el cual se encarga de representar en un gráfico de barras las métricas de cada modelo(figura 2.54).



Figura Representación de Metricas

### 2.6.2 Representación de los resultados(Componente Scorer)

Utilizando algunos parámetros de entrada que configura el usuario previamente antes de la ejecución del componente (figura 25). El usuario puede elegir cómo será la representación en diagrama de dispersión, de los resultados del agrupamiento con cada algoritmo por separado.

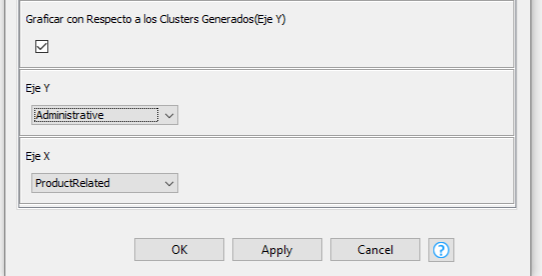


Figura Parametros de Entrada encargados de la representación

Mediante el gráfico de dispersión se pueden observar los resultados de la graficación de los datos y el comportamiento de los elementos de cada clúster. Esta respuesta del componente es crucial para el análisis de los datos (figura 26).

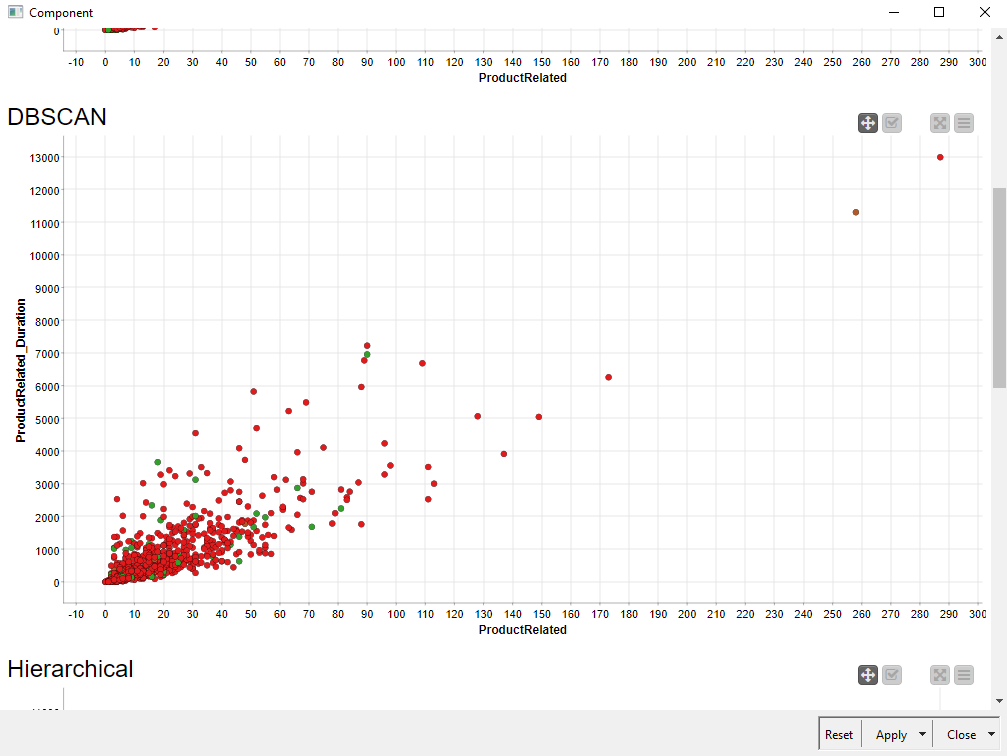


Figura Representación de Resultados

## Conclusiones Parciales

Recapitulando lo descrito en este Capítulo:

1.  **Mecanismo de Pre-procesado**: Se describe un enfoque detallado para preparar los datos antes de aplicar los algoritmos de agrupamiento, haciendo énfasis en la recolección de datos, limpieza de valores únicos y manejo de valores faltantes. Este pre-procesamiento asegura que los datos tengan la calidad requerida para evitar errores de compilación y mejorar la precisión de los modelos.
2.  **Prototipo del Componente AutoML Clustering**: Se presenta el desarrollo de un componente general de AutoML en KNIME, que incluye módulos para optimizar los hiperparámetros de algoritmos como K-Means, DBSCAN y Hierarchical Clustering. Este componente permite automatizar la configuración de los modelos y seleccionar el mejor desempeño basado en métricas de calidad como el Coeficiente Silhouette.
3.  **Optimización y Procesamiento de Algoritmos de Agrupamiento**: El capítulo detalla la optimización específica para cada uno de los algoritmos de agrupamiento (K-Means, DBSCAN, Hierarchical), explicando las configuraciones de hiperparámetros necesarias y los flujos de trabajo empleados para mejorar la eficacia de cada método.
4.  **Post-Procesamiento y Representación de Resultados**: Se subraya la importancia de la fase de post-procesado, donde se puntúan y visualizan los resultados obtenidos de cada modelo. El componente Scorer se utiliza para evaluar la efectividad de los algoritmos de agrupamiento, facilitando la comparación entre modelos y garantizando que se selecciona el más adecuado para el conjunto de datos analizado.

# Conclusiones

El presente trabajo ha logrado desarrollar un componente de AutoML en KNIME específicamente diseñado para tareas de agrupamiento, lo que representa una contribución significativa al campo de la minería de datos y al uso de técnicas de aprendizaje automático no supervisado. A través de un enfoque modular, se han integrado distintas técnicas de pre-procesamiento, optimización de hiperparámetros, y representación de resultados, facilitando así la aplicación eficiente de los algoritmos de agrupamiento más utilizados: K-Means, DBSCAN y Hierarchical Clustering.

En el proceso de desarrollo, se identificaron y aplicaron mejoras clave en la preparación de los datos, un paso crítico que influye directamente en la calidad de los resultados de los algoritmos de agrupamiento. La inclusión de técnicas automatizadas para la recolección, limpieza y normalización de datos, así como el manejo efectivo de valores faltantes, ha demostrado ser fundamental para asegurar la precisión y robustez del modelo.

Además, la implementación de un componente capaz de optimizar hiperparámetros permite no solo reducir el tiempo requerido para entrenar modelos, sino también garantizar que el rendimiento sea óptimo al seleccionar automáticamente los mejores parámetros de acuerdo a las características de cada conjunto de datos. Esto se traduce en un ahorro significativo de recursos y en una mayor accesibilidad para usuarios sin un conocimiento profundo en el ajuste manual de estos parámetros.

El uso de KNIME como plataforma para este desarrollo ha mostrado múltiples ventajas, tales como su entorno intuitivo y visual, que permite a los usuarios construir y modificar flujos de trabajo con facilidad. La reutilización de componentes y la capacidad de personalización facilitan la creación de soluciones adaptadas a necesidades específicas, promoviendo la colaboración y el intercambio de conocimiento dentro de la comunidad de ciencia de datos.

El componente AutoML desarrollado ofrece una solución integral que automatiza y optimiza el proceso de agrupamiento en tareas de minería de datos. Su capacidad para manejar grandes volúmenes de datos, su enfoque flexible y modular, y su integración en KNIME lo posicionan como una herramienta valiosa tanto para investigadores como para profesionales de la industria. A medida que continúen evolucionando las técnicas y herramientas de AutoML, este trabajo proporciona una base sólida sobre la cual se pueden construir futuros desarrollos, con el potencial de expandir aún más las capacidades del aprendizaje automático automatizado en la minería de datos.

# Bibliografía

1. Lukas. Tuggener, M.A., Automated Machine Learning in Practice: State of the Art and Recent Results, in 6th Swiss Confederence on Data Science. 2019. p. 6.
2. Marc-Andr´e Z¨oller, M.F.H., Benchmark and Survey of Automated Machine Learning Frameworks. Journal of Artificial Intelligence Research, 2021. 70: p. 64.
3. Giuseppe, B., Machine Learning Algorithms. 2017: Packt Publishing Ltd.
4. Claude. Sammut, G.W., Encyclopedia of Machine Learning. 2011: Springer.
5. Gabriel Mauricio Martínez-Toro, D.R.-B., Efrén Romero-Riaño, Paola Andrea Romero-Riaño, Unsupervised learning: application to epilepsy. Revista Colombiana de Computación, 2019. 20(2): p. 7.
6. Flach, P., The Art and Science of Algorithms that Make Sense of Data. 2012: Cambridge University Press.
7. Oludare I. Abiodun, A.J., Abiodun E. Omolara, Kemi V. Dada, Nachaat A. Mohamed, Humaira Arshad State-of-the-art in artificial neural network applications: A survey. Elsevier Ltd, 2018.
8. Frank. Hutter, L.K., Joaquin. Vanschoren, Automated Machine Learning. 2019: Springer Nature.
9. Matthias. Feurer, A.K., Katharina. Eggensperger, Efficient and Robust Automated Machine Learning, in Advances in Neural Information Processing System, C.C.a.N. Lawrence, Editor. 2015.
10. Jonathan Waring, C.L., Renato Umeton, Automated Machine Learning: Review of the state-of-the-art and opportunities for healthcare. Artificial Intelligence In Medicine, 2020. 104. 71
11. Xin He, K.Z., Xiaowen Chu, AutoML: A Survey of the State-of-the-Art. 2021.
12. Anh. Truong, A.W., Jeremy. Goodsitt, Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools, in IEEE 31st International Conference on Tools with Artificial Intelligence 2019, arXiv: 1908.055572: Portland, EEUU. p. 9.
13. E. LeDell, S.P., H2O AutoML: Scalable Automatic Machine Learning, in 7th ICML Workshop on Automated Machine Learning. 2020: Viena, Austria.
14. Piali Das, N.I., Tanya Bansal, Amazon SageMaker Autopilot: a white box AutoML solution at scale, in Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning. 2020, arXiv: 2012.08483: Portland, EEUU.
15. Hernández Orallo. José, R.Q.M.J., Ferri Razmírez. César, Introducción a la Minería de Datos. 2004: Pearson Educación. 680.
16. Doris Xin, E.Y.W., Niloufar Salehi, Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows, in Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. 2021. p. 16.
17. O´Hagan. Steve, K.D. Software review: the KNIME workflow environment and its applications in genetic programming and Machine Learning. Springer Science+Business Media, 2015.
18. Syed Muzamil Basha, K.B., C. Ramesh, Robbi Rahim, R. Manikandan, Ambeshwar Kumar, Comparative Study on Performance of Document Classification Using Supervised Machine Learning Algorithms: KNIME. International Journal on Emerging Technologies, 2019. 10(1): p. 6.
19. Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. (2013). Regression: models, methods and applications. Springer Science & Business Media. <https://doi.org/10.1007/978-3-642-34333-9>
20. Malisiewicz, T., et al. (2017). Stacked Generalization. En Encyclopedia of Machine Learning and Data Mining (p. 1135-1136). Springer.
21. Painsky A, Rosset S (2017). "Cross-Validated Variable Selection in Tree-Based Methods Improves Predictive Performance". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **39** (11): (p.2142–2153).
22. Bojanowski, M., Delgado-Penin, A., & Jagla, W. (2019). A Comparison of Automated Machine Learning (AutoML) Tools. arXiv preprint arXiv:1907.00909. Retrieved from <https://arxiv.org/abs/1907.00909>
23. Carrazana Ruiz, Ernesto. (2022) Componente KNIME de AutoML para pre-procesado en tareas de Clasificación. Tesis de pregrado. Facultad Informatica. CUJAE Retrieved from <https://repositorio.cujae.edu.cu/items/ec8b4935-4e66-473c-83bb-133ce4571a60>
24. KNIME AG.(2023). Documentacion Oficial de la herramienta KNIME. 8001, Zúrich, Suiza from <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html>
25. GIRALDO, Fabián Andrés; LEON, Elizabeth and GOMEZ, Jonatan. Caracterización de flujos de datos usando algoritmos de agrupamiento. *Tecnura* [online]. 2013, vol.17, n.37 [cited 2024-08-23], pp.153-166. Available from: <http://www.scielo.org.co/scielo.php?script=sci\_arttext&pid=S0123-921X2013000300014&lng=en&nrm=iso>. ISSN 0123-921X.
26. Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. 2024. A Survey on AutoML Methods and Systems for Clustering. ACM Trans. Knowl. Discov. Data 18, 5, Article 120 (June 2024), 30 pages. https://doi.org/10.1145/3643564