Facultad de Ingeniería Informática
Universidad Tecnológica de La Habana
José Antonio Echeverría
cujae

Informe de prácticas profesionales

Modificación de componente KNIME de AutoML en tareas de clasificación.

Autores:
Jennifer Yanez Jiménez
Rainer Pellerano Alvarez

Tutora:
Dra. Raisa Socorro Llanes

La Habana, Abril 2023

Resumen

Palabras claves:

Abstract

Keywords:

Índice general

Índice de tablas

Índice de figuras

Introducción

En la actualidad, vivimos en un mundo donde cada vez más datos se generan y 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 informadas? Para resolver este reto, se han desarrollado diversas técnicas y herramientas para extraer información útil de grandes cantidades de datos. Uno de estos enfoques es el Aprendizaje Automático.

El Aprendizaje Automático, mayormente conocido como Machine Learning, es un subcampo de la Inteligencia Artificial, que se enfoca en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender de los datos sin ser programadas explícitamente para hacerlo. Se define como un conjunto de métodos que puede detectar patrones en los datos automáticamente, y luego usar los patrones descubiertos para predecir datos en el futuro, o para realizar otro tipo de toma de decisiones bajo incertidumbre (Murphy, 2012). Estos patrones interesantes son los que representan el conocimiento. Sin embargo, el proceso de aprendizaje automático requiere de un gran volumen de datos y, a menudo, estos datos no están estructurados. Es aquí donde la minería de datos entra en juego.

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). Como parte del proceso de descubrimiento de conocimiento en los datos (KDD, por sus siglas en inglés), involucra las fases de dicho proceso: integración, recopilación, selección, limpieza, transformación, evaluación e interpretación de de los datos; así como la difusión y uso del conocimiento obtenido (Jiawei Han, 2011).

La implementación efectiva de estas técnicas requiere la intervención humana, incluyendo la selección de algoritmos adecuados, el preprocesamiento de datos y la optimización de hiperparámetros. La automatización del aprendizaje automático (AutoML) se ha 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 realizar la minería de datos de manera efectiva, se requiere de herramientas especializadas que permitan procesar grandes cantidades de información de forma rápida y eficiente. Una de estas herramientas es KNIME, un software de código abierto que ofrece una amplia variedad de funciones para la preparación de datos, análisis estadísticos y

visualización de resultados.
KNIME es una herramienta popular que proporciona un entorno de desarrollo visual para la creación, ejecución y evaluación de flujos de trabajo de análisis de datos. Es una plataforma de software libre y abierto que incluye una amplia variedad de nodos para el preprocesado de datos, la minería de datos y la modelización de datos (**?**). En KNIME, el AutoML se puede implementar a través de diferentes nodos, por ejemplo, el nodo *AutoML Learner*, que permite seleccionar automáticamente el mejor algoritmo de aprendizaje automático para un conjunto de datos en particular, así como ajustar los hiperparámetros del modelo. No obstante, una de las principales desventajas que presenta es la falta de control y personalización. Aunque el nodo AutoML permite a los usuarios seleccionar automáticamente los algoritmos y ajustar los hiperparámetros, los usuarios no tienen control directo sobre estos procesos y no pueden ajustar los parámetros de manera manual. Esto puede limitar la capacidad de los usuarios para personalizar y optimizar los modelos para satisfacer sus necesidades específicas.
En (Carrazana Ruiz, 2022) se desarrolla un componente de KNIME, que ejecuta y compara el desempeño de múltiples flujos de AutoML en tareas de clasificación, partiendo de la problemática de que no es posible, mediante un único nodo, desarrollar todo el proceso de KDD con la posibilidad de visualizar y analizar las consideraciones realizadas durante el pre-procesado. En este componente se desarrollaron subcomponentes enfocados en tareas concisas de pre-procesado: el procesamiento de datos numéricos, *string*, valores faltantes y el ajuste de tipos columna; y se demostró el correcto funcionamiento del componente propuesto y los subcomponentes que lo integran. Sin embargo quedaron tareas pendientes, como la optimización de hiperparámetros y la automatización de algunas de las actividades esenciales en el pre-procesado, siendo esta la **situación problemática**.
Se define como **problema a resolver** ¿cómo incorporar a este componente la posibilidad de optimizar los hiperparámetros y algunas de las actividades iniciales en el pre-procesado a partir de un algoritmo de clasificación? Para dar solución a esta problemática, se determina como **objetivo general** desarrollar una nueva versión del componente KNIME que permita la selección de manera automatizada de valores e hiperparámetros predefinidos en la implementación anterior. Este objetivo general se desglosa en los siguientes **objetivos específicos y tareas**:

1. Analizar el estado del arte de Machine Learning, AutoML y KNIME.
    1.1 Investigar las técnicas de Machine Learning.
    1.2 Investigar los enfoques de AutoML.
    1.3 Investigar la implementación e inclusión de nuevos componentes en KNIME.
2. Modificar múltiples flujos de AutoML en tareas de Clasificación.
    2.1 Desarrollar los subcomponentes KNIME que implementen los flujos de AutoML de pre-procesado.

2.2 Desarrollar los subcomponentes KNIME que implementen los flujos de AutoML en optimización de hiperparámetros.

2.3 Desarrollar componentes KNIME para la visualización de los resultados.

3. Realizar la validación de cada flujo implementado.

3.1 Diseñar y ejecutar las pruebas y experimentos de los flujos implementados.

3.2 Documentar los resultados obtenidos.

El **valor práctico** de este proyecto consiste en las modificaciones desarrolladas al componente KNIME de AutoML para el pre-procesado en tareas de clasificación, capaz de automatizar la optimización de hiperparámetros y ejecutar flujos de pre-procesado para diferentes algoritmos en dicha tarea.
En cuanto a la estructuración, este trabajo está dividido en cuatro capítulos:

- **Capítulo 1: Aprendizaje automático y AutoML**, se presenta el estudio realizado sobre las temáticas que aborda el trabajo, se muestran los conceptos fundamentales relacionados con el Aprendizaje automático, la Minería de Datos y AutoML; así como la descripción del componente KNIME de AutoML para pre-procesado.
- **Capítulo 2: Propuesta de modificación al componente KNIME de AutoML para pre-procesado**, se presenta y expone el diseño e implementación de la modificación al componente KNIME para tareas de AutoML en pre-procesado.
- **Capítulo 3: Propuesta de componente de AutoML para la optimización de hiperparámetros en tareas de clasificación**, se presenta y expone el diseño e implementación de la modificación al componente KNIME para tareas de AutoML en optimización de hiperparámetros.
- **Capítulo 4: Integración y validación de soluciones propuestas al componente de AutoML**, se muestran, comparan y analizan los resultados obtenidos de los algoritmos para diferentes configuraciones.

Capítulo 1

Aprendizaje Automático y *AutoML*

En este primer capítulo se aborda el marco teórico de la tesis, el cual se enfoca en diferentes temas clave dentro del campo de la inteligencia artificial y la minería de datos. En particular, se discute el aprendizaje automático, el descubrimiento de conocimiento en bases de datos, la minería de datos y la clasificación. Además, se introduce el concepto de AutoML, como herramienta para automatizar el proceso de preprocesado y optimización de hiperparámetros en la implementación de técnicas de aprendizaje automático. Por último, se destaca la importancia de la plataforma KNIME como una herramienta útil para la implementación de técnicas de AutoML y análisis de datos.

1.1. Aprendizaje Automático

Uno de los campos más destacados dentro de la IA es el Machine Learning (aprendizaje automático), que es un enfoque que utiliza algoritmos y modelos matemáticos para permitir que los sistemas aprendan de los datos y realicen tareas específicas sin ser programados explícitamente. Se define el Aprendizaje Automático como un conjunto de métodos que pueden detectar automáticamente patrones en los datos y luego usar los patrones descubiertos para predecir datos futuros o para realizar otros tipos de toma de decisiones bajo incertidumbre (Murphy, 2012). Por lo tanto, el objetivo principal del aprendizaje automático es estudiar, diseñar y mejorar modelos matemáticos que se pueden entrenar (una vez o continuamente) con datos relacionados con el contexto (proporcionados por un entorno genérico), para inferir el futuro y tomar decisiones sin completo. conocimiento de todos los elementos que influyen (factores externos) (Bonaccorso, 2017).
Existen varios tipos de aprendizaje en Machine Learning, cada uno con sus propias técnicas y enfoques. A continuación, se presenta una breve descripción de algunos de los tipos de aprendizaje más comunes:

- Aprendizaje supervisado: se refiere a cualquier proceso de aprendizaje automático que aprende de una función de un tipo de entrada a un tipo de salida utilizando datos que comprenden

ejemplos que tienen valores de entrada y salida. Dos ejemplos típicos de aprendizaje supervisado son el aprendizaje de clasificación y la regresión (Sammut & Webb, 2011).

- Aprendizaje no supervisado: se refiere a cualquier proceso de aprendizaje automático que busca aprender la estructura en ausencia de un resultado identificado o retroalimentación. Tres ejemplos típicos de aprendizaje no supervisado son agrupamiento, reglas de asociación y mapas de autoorganización (Sammut & Webb, 2011).

- Aprendizaje por refuerzo: describe una gran clase de problemas de aprendizaje característicos de los agentes autónomos que interactúan en un entorno: problemas de toma de decisiones secuenciales con recompensa retrasada. Los algoritmos de aprendizaje por refuerzo buscan aprender una política (mapeo de estados a acciones) que maximice la recompensa recibida a lo largo del tiempo (Sammut & Webb, 2011).

- Aprendizaje semisupervisado: es una clase de técnicas de aprendizaje automático que hacen uso de ejemplos etiquetados y no etiquetados al aprender un modelo. En un enfoque, los ejemplos etiquetados se usan para aprender modelos de clase y los ejemplos no etiquetados se usan para refinar los límites entre clases (Jiawei Han, 2011).

En resumen, Machine Learning es una técnica que permite a las computadoras aprender patrones a partir de datos, con el objetivo de realizar predicciones o clasificaciones precisas en datos nuevos. Sin embargo, antes de aplicar técnicas de Machine Learning a un conjunto de datos, es necesario realizar una serie de procesos previos, como la selección y preprocesamiento de datos, la selección de técnicas de aprendizaje adecuadas, la evaluación y ajuste de modelos, entre otros. De esta manera, el proceso de descubrimiento de conocimiento en bases de datos complementa al Machine Learning, permitiendo una exploración más completa de los datos y una selección más adecuada de las técnicas de aprendizaje.

1.2. Proceso de descubrimiento de conocimiento en bases de datos

La Extracción de Conocimiento en Bases de Datos (*Knwodlege Discovery from Databases*, o *KDD* por sus siglas en inglés) se define como: “el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos” (Hernández Orallo *et al.*, 2004). Este proceso está compuesto por una serie de etapas o fases, descritas a continuación:

- Integración y recopilación: Se requiere poseer todos los datos que sean de utilidad a partir de las necesidades de la organización. Determina las fuentes de información que pueden ser útiles y dónde conseguirlas. Como parte del desarrollo de esta fase es necesario diseñar

o conocer el modo en que se van a organizar e integrar los datos; con el fin de eliminar redundancias e inconsistencias.

- Selección, limpieza y transformación: Se seleccionan los datos más relevantes y que aporten mejor información, garantizando que el dato tenga la mejor calidad posible logrando obtener las vistas minables, con los datos listos para la aplicación del algoritmo.
- Algoritmos de Minería de datos: A través de la vista minable obtenida en la fase anterior se aplican los algoritmos de extracción del conocimiento.
- Evaluación e Interpretación: El objetivo de esta etapa es medir la calidad de los modelos obtenidos utilizando diferentes métricas de calidad. Las cuales dependen de las técnicas de minería de datos que se utilicen. La interpretación de los resultados se apoya en el uso de técnicas de visualización y de representación con el fin de entender mejor el conocimiento aportado.
- Difusión y uso: En esta etapa, se integra el conocimiento obtenido de la comprensión del negocio, con el conocimiento de los modelos de minería de datos usado en la toma de decisiones de los especialistas. La monitorización de los patrones debe realizarse, pues en ocasiones resulta necesaria la reevaluación del modelo, su reentrenamiento o incluso su reconstrucción total.

Tras el aumento de grandes volúmenes de información en toda institución donde se almacenen datos históricos, ahora informatizados en bases de datos digitales, es necesaria la extracción de información valiosa a través de patrones ocultos en estos datos. Sin embargo, esta cantidad de información sobrepasa las capacidades de los analistas de estas instituciones, provocando la necesidad del empleo de técnicas automatizadas. De aquí, surge como una nueva necesidad la minería de datos, siendo parte del proceso KDD.

1.3. Minería de Datos

Según (Hernández Orallo *et al.*, 2004), la minería de datos es definida como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos.
El conocimiento extraído se puede presentar en forma de relaciones, patrones o reglas inferidos de los datos, o en forma de descripción un poco más concisa. Estos constituyen un modelo de datos analizados. Estos modelos, o tareas, se categorizan en predictivas y descriptivas (Hernández Orallo *et al.*, 2004).
En las tareas predictivas, los ejemplos están etiquetados y se emplean para estimar valores futuros o desconocidos de variables de interés. En este entorno se encuentra el aprendizaje supervisado.

En cambio, las tareas descriptivas son empleadas en el descubrimiento de propiedades de los datos examinados donde los ejemplos no se encuentran etiquetados. Aquí se pone de manifiesto el aprendizaje no supervisado. En (Hernández Orallo *et al.*, 2004) se describen las tareas de minería de datos como sigue:

- Clasificación: La clasificación se encarga de examinar las características de un registro u objeto, y de esta forma asignarle una clase predefinida. Estas clases son valores discretos. Para ello, se tiene que construir un modelo a partir de datos previamente clasificados. Como variantes a la clasificación, existe el aprendizaje de "rankings", aprendizaje de preferencias y el aprendizaje de probabilidad, entre otros.

- Regresión: A diferencia de la clasificación, el valor a predecir es numérico. Consiste en aprender una función real que calcula un valor para un atributo real. Su objetivo es minimizar el error entre el valor predicho y el valor real.

- Correlaciones: Son empleadas para examinar el grado de similitud de los valores de dos variables numéricas. Se basa en el cálculo de correlación de variables numéricas usando la estadística. Este método trata de determinar si el comportamiento de dos variables numéricas está relacionado.

- Reglas de asociación: Las reglas de asociación son situaciones o características que ocurren en un mismo instante de tiempo. Pueden ser relaciones causales o casuales. Representan patrones de comportamiento entre los datos en función de la aparición conjunta de valores de dos o más atributos. Las medidas habituales propuestas en (Agrawal *et al.*, 1993) para establecer la idoneidad y el interés de una regla de asociación son la confianza y el soporte.

- Agrupamiento (Clustering): Para realizar esta tarea se parte de datos sin clasificar, teniendo como objetivo segmentar un grupo de datos diversos en subgrupos. Los miembros de cada grupo (clúster, por su definición en inglés) deben tener mucho en común entre sí y, a su vez, diferenciarse del resto de elementos de otros grupos. Dado que la clasificación de estos grupos no se conoce previamente, es el minero el encargado de darles un significado.

1.3.1. Clasificación

En el uso común, la palabra clasificación significa colocar las cosas en categorías, agruparlas de alguna manera útil. Nosotros, como humanos, generalmente hacemos esto porque las cosas en un grupo, llamado *clase* en aprendizaje automático, comparten características comunes (Sammut & Webb, 2011).

En aprendizaje automático, la clasificación, se utiliza para identificar a qué clase o categoría pertenece una determinada observación o registro, basándose en un conjunto de características

o variables. En esta, se utiliza un algoritmo para construir un modelo predictivo que asigne una etiqueta de clase a cada observación en función de sus características. Este modelo se entrena utilizando un conjunto de datos etiquetados previamente, y luego se aplica a nuevos datos para hacer predicciones (Sammut & Webb, 2011).

Clasificación con Árboles de Decisión

La clasificación con árboles de decisión es un método popular en la minería de datos y en el aprendizaje automático supervisado, se utiliza para predecir la clase o categoría de un objeto o registro. Los Árboles de Decisión son unos de los modelos más populares, su representación es de fácil entendimiento, incluso por personas no afines al área, pues su construcción en sencilla: las hojas toman los valores objetivos, mientras los atributos y sus posibles valores conforman los nodos y ramas respectivamente (Sammut & Webb, 2011).

Basados en árboles de decisión existen otros algoritmos de clasificación como: ID3, C4.5 y CART. Cada uno de ellos fue desarrollado como una versión mejorada del anterior, pero todos tienen una alta eficiencia y tiempos de ejecución reducidos, lo que los hace igualmente populares en la actualidad (Javed Mehedi Shamrat *et al.*, 2022). Se comparan y analizan en (Gupta *et al.*, 2017), donde se arrojan sus principales características:

- ID3 (Iterative Dichotomiser 3): Basa su funcionamiento en la entropía y la ganancia de información. El árbol se construye iterativamente de arriba hacia abajo, eligiendo en cada caso el atributo con mayor ganancia de información, hasta que la información ganada sea cero o se haya llegado a todas las hojas (Javed Mehedi Shamrat *et al.*, 2022), (Gupta *et al.*, 2017). Maneja datos nominales y no tolera valores faltantes.
- C4.5 (Classification 4.5): Desarrollado con el objetivo de mejorar los defectos de ID3. Añade la poda, desestimando las ramas sin aportes, que reduce los errores al clasificar como resultado de un gran número de ramas en el modelo (Sammut & Webb, 2011), (Javed Mehedi Shamrat *et al.*, 2022), (Gupta *et al.*, 2017). Adicionalmente maneja valores faltantes y numéricos.
- CART (Classification and Regression Trees): Genera un árbol binario siguiendo el mismo enfoque entrópico que ID3, pero empleando el coeficiente de Gini como criterio de selección (Javed Mehedi Shamrat *et al.*, 2022), (Gupta *et al.*, 2017). Es capaz de manejar datos faltantes, al igual que datos numéricos y nominales.

Clasificación con Redes Neuronales

Las redes neuronales o redes neuronales artificiales, son algoritmos de aprendizaje basados en una vaga analogía de cómo funciona el cerebro humano. El aprendizaje se logra ajustando los

pesos en las conexiones entre nodos, que son análogas a las sinapsis y las neuronas (Sammut & Webb, 2011).
Las primeras redes neuronales conocidas como pre-alimentadas, como la de la figura 1.1, se caracterizan por tener una arquitectura en la que cada capa de neuronas está conectada completamente con la capa siguiente, pero no con la capa anterior. Esto significa que la información fluye de forma unidireccional, sin retroalimentación (Abiodun *et al.*, 2018).

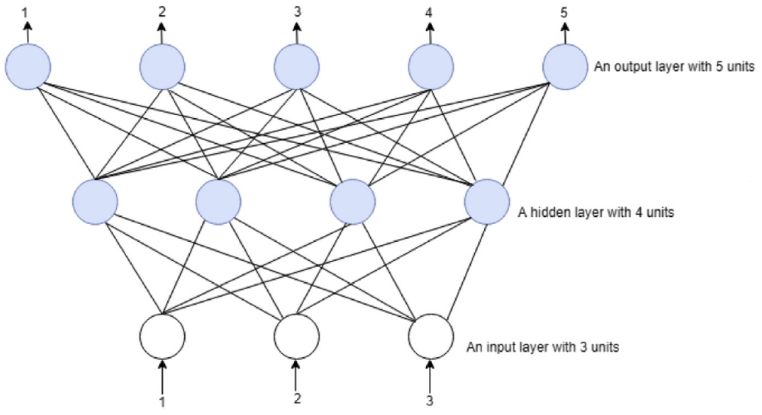

Figura 1.1: Red Neuronal Pre-Alimentada (Abiodun *et al.*, 2018)

Posteriormente, se desarrollaron las redes neuronales por retro propagación, como la presente en la figura 1.2, que comparan la salida obtenida por la red con la salida deseada, y ajustan los pesos de las conexiones entre las neuronas de la red para reducir el error de predicción. La retro propagación se utiliza para calcular la contribución de cada peso en el error de la red, y así ajustarlos de manera adecuada (Abiodun *et al.*, 2018).

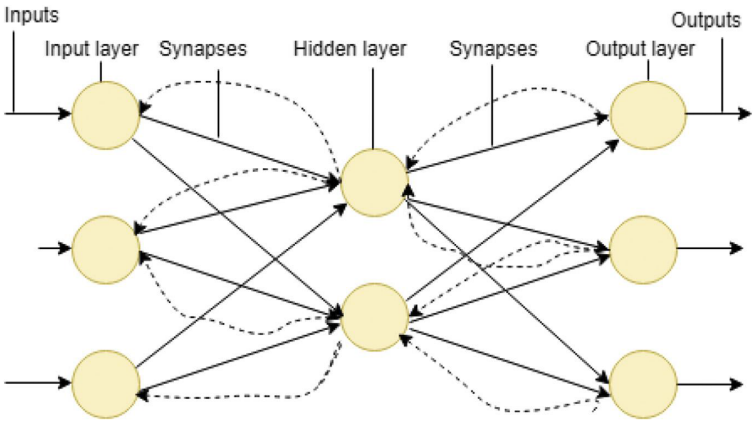

Figura 1.2: Red neuronal por retro-propagación (Abiodun *et al.*, 2018)

Gracias a su gran versatilidad se pueden aplicar en una amplia variedad de campos y discipli-

nas para resolver problemas complejos de aprendizaje automático, como es el procesamiento del lenguaje natural, reconocimiento de voz e imágenes (Abiodun *et al.*, 2018). Debido a la complejidad de sus modelos puede ser difícil su entendimiento, por lo que preferiblemente se utilizan en el contexto del reconocimiento de patrones, en la figura 1.3 se muestra un ejemplo de Red neuronal multicapa para el reconocimiento facial.

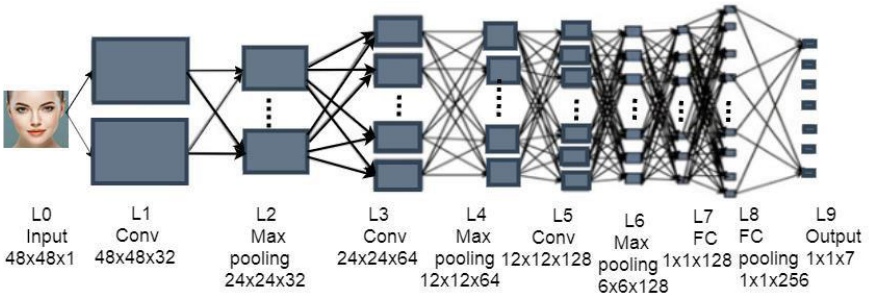

Figura 1.3: Red neuronal multicapa compleja (Abiodun *et al.*, 2018)

Clasificación con Support Vector Machine

Las Máquinas de Soporte Vectorial o Support Vector Machine (SVM) en inglés, son una clase de algoritmos lineales que se pueden usar para clasificación (Sammut & Webb, 2011), cuyo objetivo es encontrar el hiperplano que mejor separa dos clases de datos en un espacio de alta dimensionalidad (Guenther & Schonlau, 2016). En la figura 1.4 se representa como el hiperplano H2 divide con mayor margen las clases que el hiperplano H1.

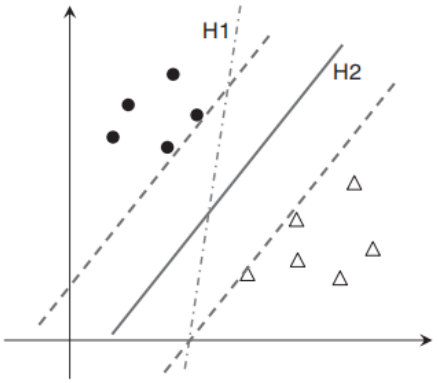

Figura 1.4: Ejemplos de posibles hiperplanos de SVM (Sammut & Webb, 2011)

Uno de los desafíos de SVM en problemas de clasificación de múltiples clases es cómo manejar la predicción de estas. En este contexto, existen dos enfoques principales:

- Uno contra uno (One-vs-One): Durante la fase de predicción, cada clasificador binario vota por la clase que cree que es correcta y la clase con el mayor número de votos se selecciona como la clase final para el punto de datos dado. Este enfoque presenta la ventaja de que cada clasificador binario solo necesita ser entrenado en un subconjunto de los datos, lo que puede ser útil cuando hay grandes conjuntos de datos. Es más resistente a desequilibrios en la distribución de clases que otros enfoques de SVM (Guenther & Schonlau, 2016).

- Uno contra todos (One-vs-All): Entrena un clasificador binario para cada clase posible, donde el conjunto de datos de una clase se considera positivo y los datos de las otras clases se consideran negativos. Durante la fase de predicción, cada clasificador binario vota por su clase correspondiente y la clase con la mayor puntuación se selecciona como la clase final para el punto de datos dado. Fácil de implementar y puede funcionar bien en conjuntos de datos pequeños, pero puede ser menos preciso en conjuntos de datos grandes y complejos (Guenther & Schonlau, 2016).

Agregar que existen otros dos enfoques: Clasificación jerárquica (Hierarchical classification) y Clasificación por parejas (Pairwise classification). Cada uno de estos tiene sus propias ventajas y desventajas, y la elección de uno de ellos dependerá del tipo de datos y del problema que se esté tratando de resolver.

1.4. *AutoML*

El campo del aprendizaje automático automatizado (AutoML) tiene como objetivo tomar 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.
AutoML (Automated Machine Learning) es una técnica que tiene como objetivo automatizar todo o parte del proceso de Machine Learning, incluyendo la selección de algoritmos, la optimización de hiperparámetros, la selección de características y la evaluación del rendimiento del modelo. La relación entre AutoML y Machine Learning es que AutoML es una técnica que se utiliza para automatizar el proceso de Machine Learning, lo que significa que se utiliza para automatizar todo o parte del proceso de selección del mejor modelo de Machine Learning para un conjunto de datos dado. La automatización del proceso de Machine Learning proporciona una solución eficiente y escalable para el análisis de grandes conjuntos de datos, lo que puede resultar en un ahorro significativo de tiempo y recursos para los profesionales de ciencia de datos y los investigadores.

1.4.1. Preprocesado

.....

Discretización

.....

1.4.2. Optimización de hiperparámetros

Cada sistema de aprendizaje automático tiene hiperparámetros, y la tarea básica de AutoML es automatizar el ajuste de estos para maximizar el rendimiento del modelo en un conjunto de datos de prueba o validación (Hastie *et al.*, 2009). La optimización de hiperparámetros automatizada (HPO) tiene varios casos de uso importantes (Hutter *et al.*, 2019); puede

- reducir el esfuerzo humano necesario para aplicar el aprendizaje automático. Particularmente importante en el contexto de AutoML.
- mejorar el rendimiento de los algoritmos de aprendizaje automático (adaptándolos al problema en cuestión).
- mejorar la reproducibilidad y equidad de los estudios científicos.

Para ahorrar tiempo y mejorar la precisión de los modelos de aprendizaje automático se combina la selección de algoritmos y la optimización de hiperparámetros en un solo proceso, denominado Selección de Algoritmos y Optimización de Hiperparámetros Combinados (CASH por sus siglas en inglés) (Tuggener *et al.*, 2019).
CASH en conjunción con la automatización del pre-procesado de los datos, integran el problema general del AutoML, reflejado en la figura 1.5.

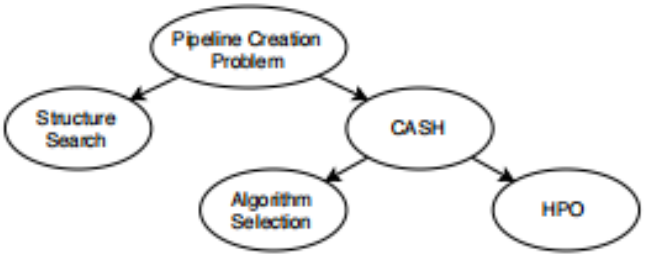

Figura 1.5: Desglose de los subproblemas de AutoML (Zöller & Huber, 2021)

Existen varias estrategias de HPO, algunas de las cuales son:

- Búsqueda aleatoria: Selecciona valores de forma aleatoria dentro de un rango definido. No garantiza encontrar los mejores valores posibles y puede requerir numerosas iteraciones para encontrar un conjunto de hiperparámetros que proporcione un rendimiento óptimo (Géron, 2022), (Zöller & Huber, 2021).

- Búsqueda voraz: Prueba todas las combinaciones posibles de valores de los hiperparámetros dentro de un rango definido. Presenta una alta demanda computacional, imposible de manejar a medida que escalan los sistemas y las bases de datos (Zöller & Huber, 2021).

- Búsqueda en cuadrícula: Prueba cada combinación de valores de hiperparámetros y selecciona la combinación de hiperparámetros que ha dado el mejor rendimiento. Puede ser computacionalmente costoso si el espacio de búsqueda y el número de hiperparámetros es grande (He *et al.*, 2021).

- Optimización Bayesiana: Construye modelos probabilísticos para representar la función objetivo, que se actualiza después de cada iteración. Maneja espacios de búsqueda de alta dimensionalidad y no requiere tantas iteraciones como la búsqueda en cuadrícula o la búsqueda aleatoria para encontrar combinaciones de hiperparámetros de alto rendimiento (Hutter *et al.*, 2019), (He *et al.*, 2021).

- Optimización basada en gradiente: Emplea la información del gradiente para optimizar los hiperparámetros de manera iterativa. Genera una gran carga computacional y presenta la limitación de caer en mínimos locales (Zöller & Huber, 2021).

A pesar del incesante estudio vinculado al HPO, este sigue presentando en la actualidad diversos desafíos (Hutter *et al.*, 2019):

- Costo computacional: la optimización de hiperparámetros puede ser muy costosa en términos de tiempo de cómputo y recursos de hardware, especialmente cuando se utiliza un espacio de hiperparámetros grande o se ejecutan muchas iteraciones de entrenamiento. Esto puede limitar la escalabilidad y la eficiencia de la HPO.

- La generalización del modelo: la optimización de hiperparámetros puede resultar en un modelo altamente ajustado que no generaliza bien a nuevos datos. Se torna complejo cuando las bases de datos poseen múltiples tipos de datos.

- Complejidad del espacio de hiperparámetros: el espacio de hiperparámetros puede ser muy complejo y estar altamente interconectado, lo que dificulta la exploración y la selección de los hiperparámetros adecuados.

Entre las aplicaciones de HPO se pueden encontrar (Hernández & Ortiz-Hernández, 2017), donde se aplica HPO en SVM y Bosque aleatorio para la Predicción de Enfermedades Cardiovasculares y (Waring *et al.*, 2020), que manifiesta el desarrollo del problema de HPO en redes Neuronales en un contexto de análisis de salud.

1.5. Herramienta de minería de datos KNIME

En la minería de datos se utilizan diferentes herramientas que simplifican el trabajo. KNIME es una de las principales herramientas de minería y análisis de datos, siendo muy completa y ofreciendo muchas funcionalidades (Ilastegui, 2012).
La herramienta de datos KNIME (*Konstanz Information Miner*, por sus siglas en inglés), es una plataforma de minería de datos de código abierto, disponible para varias plataformas y sistemas operativos, que permite el desarrollo de modelos en un entorno visual. Esta herramienta tiene como objetivo desarrollar procesos de KDD a través de un entorno visual. Se le considera una herramienta gráfica, ya que permite construir flujos de trabajo (**?**).
Los flujos se componen de flechas y nodos que se pueden combinar entre sí. Los nodos contienen funcionalidades tales como: algoritmos de minería de datos, formas de conexión a los datos almacenados, preprocesamiento de datos, reportes, entre otros. Las flechas indican el orden de ejecución y el flujo de la información. En la figura 1.6 se muestra un ejemplo de un flujo en KNIME para cargar y filtrar datos de una tabla, y posteriormente guardar los resultados en un fichero .csv.

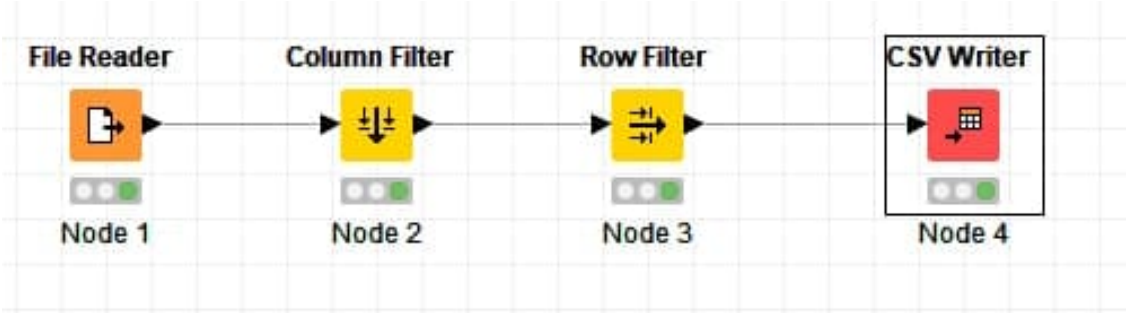

Figura 1.6: Ejemplo de flujo de trabajo en la herramienta KNIME.

La herramienta KNIME puede ser extendida a través de plugins, la mayoría son nuevos nodos, aunque las extensiones pueden ser a cualquier parte de la arquitectura. La extensibilidad de la herramienta es de forma sencilla, ya que está basada en la Plataforma de Cliente Enriquecido de Eclipse (*Eclipse RCP*, por sus siglas en inglés) (**?**). Gracias a esto, la adición de nuevos plugins a KNIME se torna menos compleja para el desarrollador.
KNIME se diseñó en base a tres principios: modularidad, extensibilidad y ambiente de trabajo interactivo. A continuación, se describen estos principios (Ilastegui, 2012):

- Modularidad: Plantea que no deben existir dependencias entre las unidades contenedoras de datos o de procesamiento. Además, se pueden implementar algoritmos de manera independiente. De igual forma, al no tener tipos de datos predefinidos, se pueden definir nuevos tipos de datos, con sus características y especificaciones propias. Estos pueden declararse compatibles con otros existentes.

- Extensibilidad de forma sencilla: Permite adicionar nuevas unidades de procesamiento, visualización y tratamiento de datos, teniendo en cuenta que esto debe ser una tarea fácil de realizar.
- Ambiente de trabajo visual e interactivo: Los flujos de trabajo deben ser fáciles e interactivos para el usuario. Por tal motivo, se harán arrastrando los elementos al área de trabajo.

.....

1.6. Componente en KNIME de AutoML para el pre-procesado en tareas de clasificación

.....

1.7. Bases de datos de prueba

.....

1.7.1. Repositorio *Kaggle*

.....

1.7.2. Repositorio *UCI Machine Learning*

.....

1.8. Conclusiones parciales

A partir de lo estudiado en este capítulo, se llega a las siguientes conclusiones:

Conclusiones

Al finalizar este trabajo se arriban a las siguientes conclusiones: .....

Recomendaciones

Con vistas a darle continuidad a este trabajo se recomienda: .....

Referencias bibliográficas

Abiodun, Oludare Isaac, Jantan, Aman, Omolara, Abiodun Esther, Dada, Kemi Victoria, Mohamed, Nachaat AbdElatif, & Arshad, Humaira. 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, **4**(11), e00938.

Agrawal, Rakesh, Imielinski, Tomasz, & Swami, Arun. 1993. Mining Association Rules between Sets of Items in Large Databases.

Bonaccorso, Giuseppe. 2017. *Machine learning algorithms*. Packt Publishing Ltd.

Carrazana Ruiz, Ernesto. 2022. Componente KNIME de AutoML para pre-procesado en tareas de Clasificación.

Géron, Aurélien. 2022. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. .ºReilly Media, Inc.".

Guenther, Nick, & Schonlau, Matthias. 2016. Support vector machines. *The Stata Journal*, **16**(4), 917–937.

Gupta, Bhumika, Rawat, Aditya, Jain, Akshay, Arora, Arpit, & Dhami, Naresh. 2017. Analysis of various decision tree algorithms for classification in data mining. *International Journal of Computer Applications*, **163**(8), 15–19.

Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome H, & Friedman, Jerome H. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer.

He, Xin, Zhao, Kaiyong, & Chu, Xiaowen. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, **212**, 106622.

Hernández, Eduardo Sánchez-Jiménez Yasmín, & Ortiz-Hernández, Javier. 2017. Técnicas de Optimización de Hiperparámetros en Modelos de Aprendizaje Automático para Predicción de Enfermedades Cardiovasculares.

Hernández Orallo, José, *et al.* 2004. *Introducción a la Minería de Datos*. Biblioteca Hernán Malo González.

Hutter, Frank, Kotthoff, Lars, & Vanschoren, Joaquin. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

Ilastegui, Lisandra Bravo. 2012. PROPUESTA DE HERRAMIENTA PARA APLICAR MINERÍA DE DATOS EN ENTORNOS COMPLEJOS.

Javed Mehedi Shamrat, FM, Ranjan, Rumesh, Hasib, Khan Md, Yadav, Amit, & Siddique, Abdul Hasib. 2022. Performance evaluation among ID3, C4.5, and CART Decision Tree Algorithm. *Pages 127–142 of: Pervasive Computing and Social Networking: Proceedings of ICPCSN 2021*. Springer.

Jiawei Han, Micheline Kamber, Jian Pei. 2011. *Data Mining. Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. 3rd edn.

Murphy, Kevin P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Sammut, Claude, & Webb, Geoffrey I. 2011. *Encyclopedia of machine learning*. Springer Science & Business Media.

Tuggener, Lukas, Amirian, Mohammadreza, Rombach, Katharina, Lörwald, Stefan, Varlet, Anastasia, Westermann, Christian, & Stadelmann, Thilo. 2019. Automated machine learning in practice: state of the art and recent results. *Pages 31–36 of: 2019 6th Swiss Conference on Data Science (SDS)*. IEEE.

Waring, Jonathan, Lindvall, Charlotta, & Umeton, Renato. 2020. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial intelligence in medicine*, **104**, 101822.

Zöller, Marc-André, & Huber, Marco F. 2021. Benchmark and survey of automated machine learning frameworks. *Journal of artificial intelligence research*, **70**, 409–472.

Anexo A

Anexos

A.1. Fases y tareas de la metodología CRISP-DM

Comprensión del negocio	Comprensión de los datos	Preparación de los datos	Modelación	Evaluación	Explotación (Despliegue)
Determinar los objetivos del negocio • Trasfondo • Objetivos del negocio • Criterios de éxito del negocio **Evaluar la situación** • Listado de recursos • Requerimientos, supuestos y restricciones • Riesgos y contingencias • Terminología • Costos y beneficios **Determinar los objetivos de la minería** • Objetivos de la minería • Criterios de éxito de la minería **Elaborar el plan del proyecto** • Plan del proyecto • Valoración inicial de las herramientas y técnicas	**Recopilar los datos iniciales** • Reporte de la colección de datos iniciales **Describir los datos** • Reporte de la descripción de los datos **Explorar los datos** • Reporte de la exploración de los datos **Verificar la calidad de los datos** • Reporte de la calidad de los datos	*Salidas finales:* • Conjunto de datos • Descripción del conjunto de datos **Seleccionar los datos** • Fundamentación de la inclusión o exclusión **Limpiar los datos** • Reporte de la limpieza de los datos **Construir los datos** • Atributos derivados • Registros generados **Integrar los datos** • Datos asociados **Estructurar los datos** • Datos reconstruidos	**Seleccionar las técnicas de modelado** • Técnica de modelación • Suposiciones del modelado **Generar el diseño de experimento** • Diseño de pruebas **Construir los modelos** • Escenario de parámetros • Modelos • Descripción de los modelos **Evaluar los modelos** • Valoración de los modelos • Escenario de parámetros revisado	**Evaluar los resultados** • Valoración de la minería respecto a los criterios de éxito del negocio • Modelos aprobados **Revisar el proceso** • Revisión del proceso **Determinar los próximos pasos** • Listado de las posibles acciones • Decisión	**Planificar la explotación** • Estrategia de explotación **Planificar el monitoreo y el mantenimiento** • Plan de monitoreo y mantenimiento **Producir reportes finales** • Reporte final • Presentación final **Revisar el proyecto** • Documentación de las experiencias.

Figura A.1: Fases y tareas de la metodología CRISP-DM.

A.2. Estudio de algoritmos supervisados

Algoritmos supervisados empleados en el análisis y detección de fraude bancario

Articulo	Ámbito en que fue empleado	Tipo de técnica	Disponibilidad de implementación	Variables analizadas	Notas
Li2012	Identificar los signos de cuentas fraudulentas en ATM.	Reglas de asociación, técnicas bayesianas	KNIME	Variables del cliente: Cuentas, dirección, nombre, detalle de las transacciones. Tener en cuenta que necesita de una base de datos con transacciones clasificadas por tipo de estado (fraudulentas o no).	-
Laimek2018	Detección de fraude en ATM.	Local outlier factor (LOF), Isolation Forest	KNIME	account id, card no, transaction dom, transaction dow, trans amt, transaction cd, terminal id, terminal bank id, terminal country, transaction, velocity, fraud flag	LOF: Alto grado de complejidad. Es computacionalmente caro, ya que cada instancia de datos, tienen que explorar no sólo su densidad local, sino también la de sus vecinos. Isolation Forest: Baja complejidad de tiempo lineal y pequeño requerimiento de memoria.
Murillo2019	Fraudes en Transacciones Bancarias.	Random Forest, Naïve Bayes, J48	KNIME, excepto J48	Paso, Tipo, Monto, Nombre Origen, Viejo Balance Origen, Nuevo Balance Origen, Nombre Destino, Viejo Balance Destino, Nuevo Balance Destino ,Es Fraude, Es Bandera De Fraude	-
Gao2019	Predicción de fraude con tarjeta de crédito.	Regresión lógica, Redes neuronales, SVM, Random forest	Python, KNIME	Variables en dependencia del algoritmo.	-
Zhang2020	Detección de anomalías en datos industriales y detección de fraude en tarjetas de crédito.	SVM, Random Forest	Python, KNIME	Variables de la transacción: puntaje en cuanto al volumen, puntaje en cuanto a la velocidad en la que se realiza la transacción, puntaje en cuanto al tipo de actividad.	SVM: - Se basa en la teoría del aprendizaje estadístico. - No requiere grandes conjuntos de datos. - El entrenamiento converge a una solución única global.

Figura A.2: Algoritmos supervisados estudiados dedicados a la detección de fraude bancario

A.3. Estudio de algoritmos no supervisados

Algoritmos no supervisados empleados en el análisis y detección de fraude bancario					
Articulo	**Ámbito en que fue empleado**	**Tipo de técnica**	**Disponibilidad de implementación**	**Variables analizadas**	**Notas**
Bolton2001unsupervised	Detección de fraude en tarjetas de crédito.	Análisis de Grupos Pareados y Análisis de Puntos de Ruptura.	-	-	Puede ser utilizado en conjuntos de datos muy grandes. Útil para determinar transacciones similares.
JohnSalermo2003	Detección de lavado de dinero en transacciones bancarias.	Vecino mas cercano y K-means.	KNIME	Nombre, apellido, nombre de organización, fecha y hora de transacción, direcciones de ubicación, monto de dinero.	Efectiva para descubrir conductas sospechosas analizando conductas individuales de clientes o conductas de varios grupos de clientes mediante sus transacciones. Se vuelve muy complejo para conjuntos de datos grades o largos periodos de tiempo.
Sudjianto2010	Fraude bancario y lavado de dinero	Escala multidimensional (MDS) y K-means, Analisis de Grupos Pareados (PGA)	KNIME	-	Realizan un análisis de métodos para detección de fraude donde mencionan MDS combinado con K-means, y también el método PGA de [Bolton2001unsupervised]
LIU2011	Detección de lavado de dinero en transacciones bancarias.	BIRCH y K-means	Python, KNIME	Id_transaccion, hora de transacción, persona remitente, persona receptora, frecuencia, monto de transaccion	Necesita de buenos datos para su entrenamiento. Emplean arboles de decisión para mejorar el agrupamiento. BIRCH es un algoritmo incremental apto para una gran base de datos, no puede manejar muy bien los datos financieros. BIRCH no es sensible al ruido y datos aislados que son los más importantes en la lucha contra el lavado de dinero. Para resolver esto introducen un árbol de decisiones central en el sistema basado en K-means y BIRCH.
TanmayKumarBehera2015	Detectar fraude online	Fuzzy Clustering y Redes neuronales	KNIME	Monto de transacción y artículos comprados (para el clustering) Artículos comprados y hora de transacción (para el entrenamiento de la red neuronal)	Métricas empleadas: FP, P/Sensitivity, TN/Specificity Hasta 93,9% TP y menos de 6.10% FP
Kargari2018	Detectar fraude bancario online y en cajeros	K-means (c-means), A priori, Análisis de componentes principales (PCA)	KNIME	id_transaccion, hora, número de cuenta, número de tarjeta, tipo de transacción (cajero, online), tipo de entrada (tarjeta presente/no presente), monto de transacción, código mercado, grupo mercado, genero dueño tarjeta, edad dueño tarjeta, banco.	Se propone un modelo semi supervisado aplicando clustering y reglas de asociación. Los resultados de este modelo son más exactos que cada una de las reglas de agrupación y asociación por separado.

Figura A.3: Algoritmos no supervisados estudiados dedicados a la detección de fraude bancario

A.4. Modelo de Árbol de Decisión

A.4.1. Resultados de la ejecución del nodo *Decision Tree*

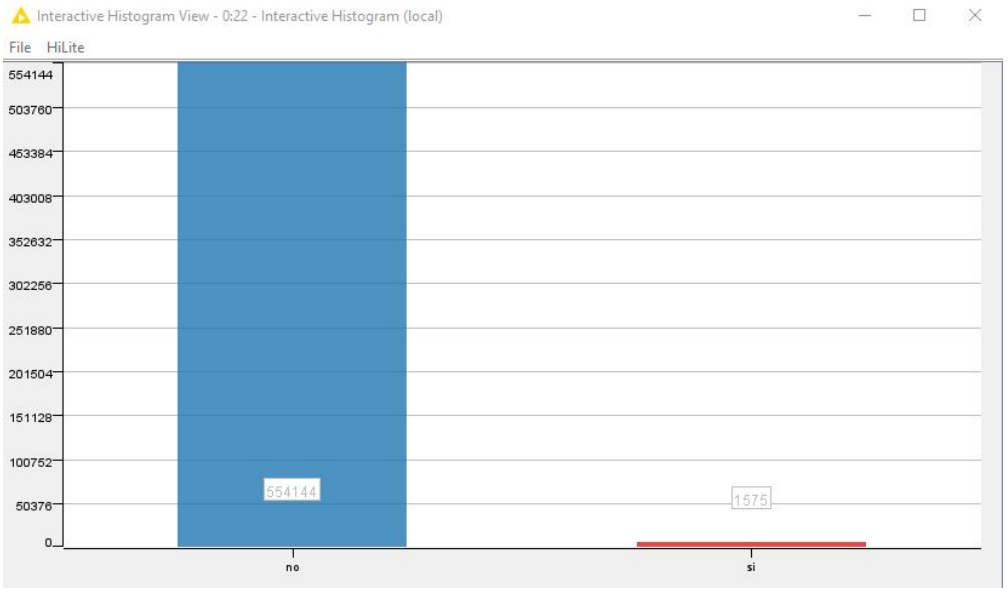

Figura A.4: Resultados del nodo *Decision Tree* con todas las variables de la vista minable

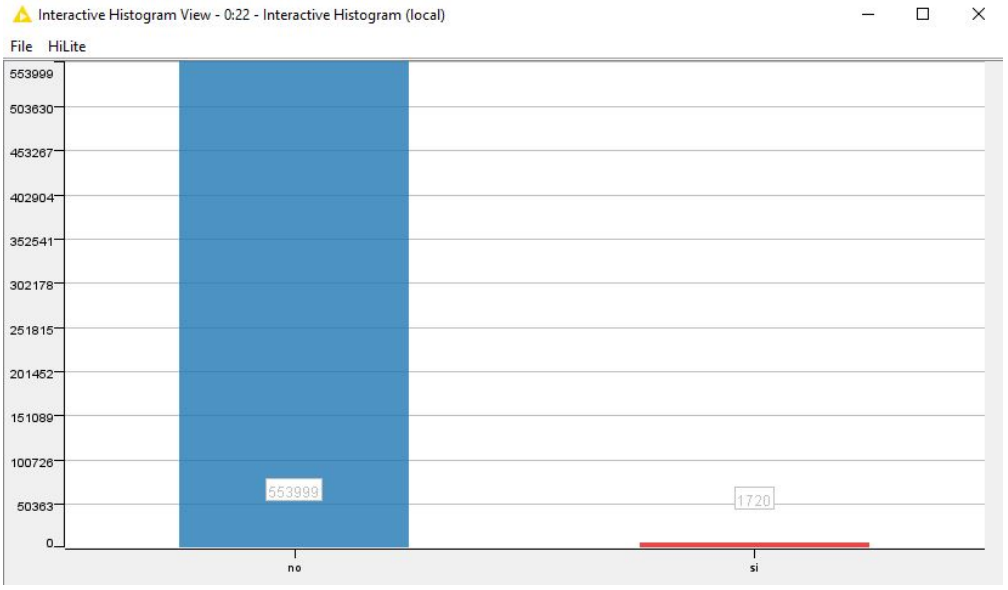

Figura A.5: Resultados del nodo *Decision Tree* tras la elección de variables

A.5. Modelo de Random Forest

A.5.1. Resultados de la ejecución del nodo *Random Forest*

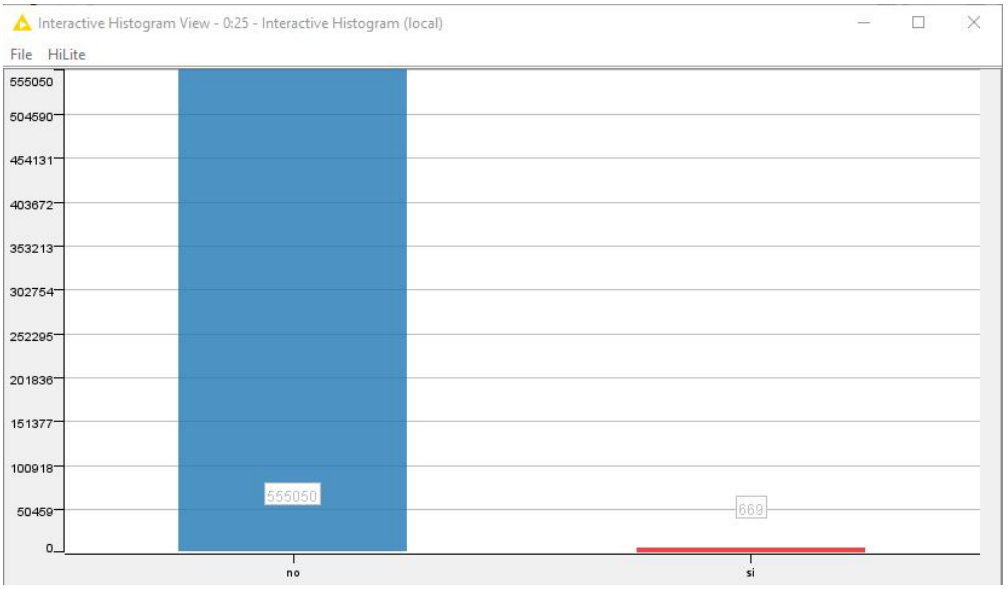

Figura A.6: Resultados del nodo *Random Forest* con todas las variables de la vista minable

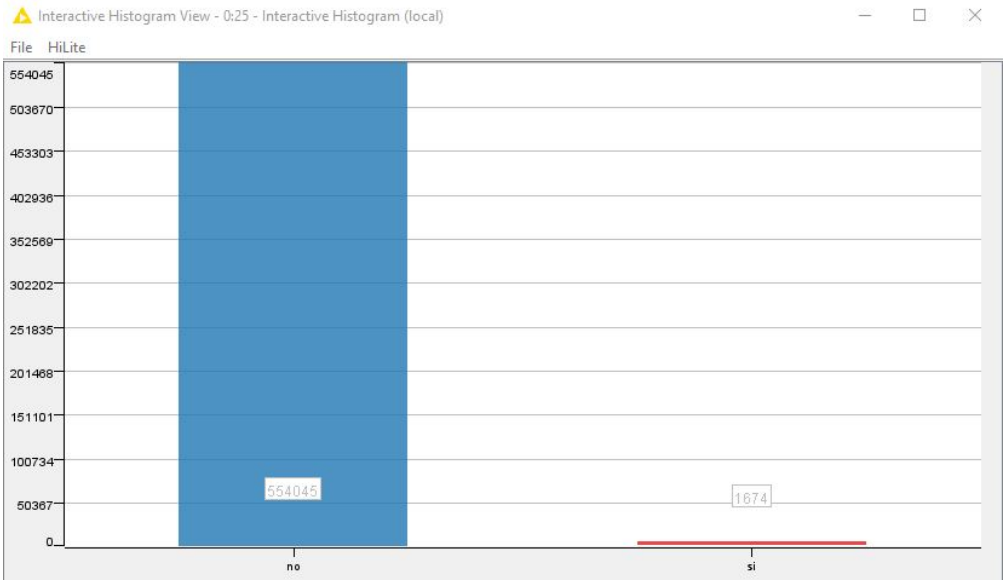

Figura A.7: Resultados del nodo *Random Forest* tras la elección de variables