ELEMENTOS BÁSICOS DE ANÁLISIS INTELIGENTE DE DATOS

AUTORES:
Iván Fredy Jaramillo Chuqui
Ricardo Villarroel Molina

www.editorialgrupo-aea.com

Elementos básicos de Análisis Inteligente de Datos

Autor/es:

Jaramillo-Chuqui, Iván Fredy

Villarroel-Molina, Ricardo

Publicado en: https://www.editorialgrupo-aea.com/
Contacto: +593 983652447; +593 985244607 **Email**: info@editorialgrupo-aea.com

Título del libro:
Elementos básicos de Análisis Inteligente de Datos

Libro Digital, Primera Edición, 2023
Editado, Diseñado, Diagramado y Publicado por Comité Editorial del Grupo AEA, Santo Domingo de los Tsáchilas, Ecuador, 2023

ISBN: 978-9942-651-20-4

https://doi.org/10.55813/egaea.l.2022.65

Como citar: Jaramillo-Chuqui, I. F., Villarroel-Molina, R. (2023). Elementos básicos de Análisis Inteligente de Datos. Primera edición. Editorial Grupo AEA. Ecuador. https://doi.org/10.55813/egaea.l.2022.65

Palabras Clave: Inteligencia, Datos, R project

Cada uno de los textos de Editorial Grupo AEA han sido sometido a un proceso de evaluación por pares doble ciego externos (double-blindpaperreview) con base en la normativa del editorial.

Revisores:

Ing. Castrejón Valdez Manuel, Ph.D.	Universidad Nacional de Huancavelica – Perú
Ing. Mencia Sánchez Noemi Gladys, Ph.D.	Universidad Nacional de Huancavelica – Perú

Los libros publicados por **"Editorial Grupo AEA"** cuentan con varias indexaciones y repositorios internacionales lo que respalda la calidad de las obras. Lo puede revisar en los siguientes apartados:

Editorial Grupo AEA

http://www.editorialgrupo-aea.com

Editorial Grupo AeA

editorialgrupoaea

Editorial Grupo AEA

Aviso Legal:

La información presentada, así como el contenido, fotografías, gráficos, cuadros, tablas y referencias de este manuscrito es de exclusiva responsabilidad del/los autor/es y no necesariamente reflejan el pensamiento de la Editorial Grupo AEA.

Índice

Índice de Tablas

Índice de Figuras

Introducción

En la era digital actual, donde los datos son generados a una velocidad exponencial, la inteligencia de datos emerge como un campo esencial para desentrañar el valor latente en estas vastas corrientes de información. Al aprovechar tecnologías avanzadas y técnicas analíticas, la inteligencia de datos constituye las herramientas para que las organizaciones no solo comprendan mejor su propia realidad, sino también anticipar tendencias, desafíos y oportunidades en un mundo empresarial en constante cambio.

Esta obra se define como un recurso para iniciar el estudio del análisis inteligente de datos, está dirigido a estudiantes que cursan carreras relacionadas a los sistemas de información y la computación.

Una herramienta prominente en este campo es R. Conocido por su poder y versatilidad en el análisis estadístico y la visualización de datos. Al aprovechar las capacidades de R, los profesionales de la información pueden explorar conjuntos de datos de manera personalizada, abriendo la puerta a una toma de decisiones más informada y estratégica en las organizaciones.

En el capítulo uno se aborda las concepciones asociadas a una exploración sistemática de información valiosa a partir de conjuntos de datos diversos y complejos. Este apartado se apoya en un sólido conjunto de fundamentos que abarcan desde la recopilación inicial de datos hasta la estructuración de los datos. A través de la aplicación de principios estadísticos, métodos matemáticos y técnicas de análisis, la inteligencia de datos en general busca descubrir patrones, correlaciones y relaciones ocultas que ofrecen una comprensión más profunda del entorno empresarial.

Antes de que los datos puedan ser analizados con precisión, es esencial someterlos a un proceso de preprocesado. El capítulo dos incluye una revisión de temas asociados a la limpieza, transformación y estructuración para eliminar errores, duplicados y valores atípicos que podrían distorsionar los resultados.

Una vez preparados, los datos pueden ser sometidos a técnicas tanto supervisadas como no supervisadas, que son abordadas en el capítulo tres y cuatro. Las técnicas supervisadas, como la regresión y la clasificación, se

explora mediante ejemplos y códigos cortos en R, sobre conjuntos de datos etiquetados para predecir valores futuros o categorizar elementos.

Por otro lado, las técnicas no supervisadas, como el agrupamiento y la reducción de dimensionalidad, exploran patrones intrínsecos en los datos sin etiquetas previas, revelando agrupamientos y estructuras que a menudo pasan desapercibidos, también se presenta con un enfoque teórico práctico para los métodos más usados.

En conjunto, los elementos teóricos y prácticos que se ha introducido en esta obra conforman la esencia de la inteligencia de datos, una disciplina que genera valor agregado en los profesionales de la informática hoy en día.

01 CAPITULO

FUNDAMENTOS DE ANÁLISIS INTELIGENTES DE DATOS CON R

Fundamentos de análisis inteligentes de datos con R

1.1. Introducción y objetivos

Las bases teóricas siempre constituyen el inicio de los estudios prácticos y experimentales, la inteligencia de datos está soportado en varios conceptos relacionados a la gestión de los datos, evolución de las computadoras, algoritmos, estadística, etc. La revisión de todas las teorías que anteceden a esta definición es demasiado grande, sin embargo, se pretende dar una vista general de las ciencias afines para comprender la naturaleza de este contenido.

Los textos no pueden cubrir a todas las audiencias, debido a que las construcciones temáticas parten de perfiles especializados, es así que este capítulo y los demás contenidos de este libro están dirigidos para académicos, estudiantes e investigadores que apliquen ciencia de datos mediante el programa R Project. El capítulo termina con la descripción de algunas propiedades de la herramienta R Project, y su aplicación en la inteligencia de datos.

Al finalizar este capítulo los lectores estarán en la capacidad de:

- Construir la definición de inteligencia de datos y sus sinónimos en la literatura.
- Reconocer las áreas, temas y ciencias afines a la inteligencia de datos.
- Comprender los retos basados en la importancia y masificación de los datos en las organizaciones.
- Definir las fases que están asociados a un proceso de análisis inteligente de datos.

1.2. Origen de la inteligencia de datos

1.2.1. Antecedentes

La inteligencia de datos es un concepto que está asociado al análisis de los datos, y las personas suelen utilizar nombres como “minería de datos”,

"inteligencia de negocios", "bigdata", "descubrimiento del conocimiento", entre otros. Todas estas denominaciones abarcan en el concepto sobre analítica de los datos, las denominaciones se deben al contexto en el que se realiza el estudio.

De todas las áreas que se relacionan con el análisis inteligente de datos, la *estadística* es la más antigua, sin embargo, estas dos disciplinas no comparten el mismo origen. Los principios estadísticos de acuerdo a lo que se conoce hoy en día tienen sus orígenes desde la época AC (Antes de Cristo), con tareas asociadas al registro desde los censos poblacionales, definiciones relacionadas a la probabilidad, la publicación del teorema de Bayes en el siglo XVII, los conceptos de estadística inferencial en el siglo XVIII con temas de coeficientes de correlación y regresión lineal (Gupta & Parsad, n.d.).

Los conceptos de regresión lineal, probabilidades (Teorema de Bayes) y los estudios correlacionales han marcado el inicio del estudio de los datos en una polaridad inferencial. El surgimiento de la computadora que introdujo el registro y organización de los datos en un sistema de codificación binario, esto unido a la capacidad de cálculo, permitió el desarrollo de los modelos matemáticos y estadísticos mediante un programa de computadora.

En 1943 Warren McCulloch y Walter Pitts fueron los primeros en crear un modelo conceptual de una red neuronal. En un artículo titulado Un cálculo lógico de las ideas inmanentes en la actividad nerviosa (McCulloch & Pitts, 1943), describen la idea de una neurona en una red. Cada una de estas neuronas puede hacer 3 cosas: recibir entradas, procesar entradas y generar salidas. Posteriormente en la década de los años 60 Lawrence J. Fogel formó una nueva compañía llamada *Decision Science, Inc*. para aplicaciones específicamente de la computación evolutiva en la resolución de problemas del mundo real.

En la década de 1970 con sofisticados sistemas de administración de bases de datos, es posible almacenar y consultar terabytes y pentabytes de datos. Además, los almacenes de datos permiten a los usuarios pasar de una forma de pensar orientada a las transacciones a una forma más analítica de ver los datos. Sin embargo, extraer conocimientos sofisticados de estos almacenes de datos de modelos multidimensionales es muy limitado todavía. En 1975 John Henry

Holland escribió *Adaptation in Natural and Artificial Systems*, el libro pionero sobre algoritmos genéticos. De esta forma se introduce nuevos enfoques para la resolución de problemas con complejidad difíciles.

La década de 1980 estuvo marcada por la aparición del término “minería de bases de datos” que en principio se asoció a una marca registrada, sin embargo, el proceso evolutivo de las capacidades de cálculo de las computadoras, junto con el surgimiento de más técnicas de aprendizaje máquina hicieron que surjan nuevas definiciones tales como KDD (Knowledge Discovery in Database).

A partir de 1990 las tareas que involucran a la computación evolutiva adoptan denominaciones enfocadas al contexto, así “minería de datos”, “descubrimiento del conocimiento”, “inteligencia de negocios”, etc. Nuevas variantes de aprendizaje máquina aparecen, algoritmos para: SVM (Support Vector Machine), patrones frecuentes, reglas de asociación, árboles de decisión, etc.

Es así que ya para el siglo 20, la concepción de “ciencia de datos” es visto como una disciplina independiente. El crecimiento acelerado de los volúmenes de datos en los medios de almacenamiento, debido a la proliferación de mecanismos inteligentes para la captura de datos; esto representa una tendencia que marca diversos estudios e investigaciones para trabajar con esa gran masa de datos.

1.2.2. Bases de datos

La evolución de las tecnologías digitales con la miniaturización mediante la aparición de los transistores y los circuitos integrados contribuyen progresivamente al desarrollo de software de gestión de datos. Es así que en la década de los años 60 cuando la computadora empieza a verse con fines comerciales, surge el concepto de base de datos como una forma de organización lógica de datos en un archivo (Bachman, 1972).

En la década de 1970, Edgar Codd (Codd, 1970) hace una contribución potencial al concepto de sistema manejador de bases de datos; Edgar Codd trabajó para IBM y con el propósito de facilitar el manejo de las bases de datos que hasta el momento era muy difícil, introduce a través de una investigación “A Relational Model of Data for Large Shared Data Banks” el concepto de modelo relacional,

entonces ya se habla de tablas y registros, lo que sería el comienzo de los DBMS relacionales.

Los modelos de bases de datos relacionales evolucionaron básicamente en tres dimensiones (Integridad, Consistencia y Seguridad) garantizando organización y ordenamiento en los datos. Ya en la década de 1980 muchos fabricantes de software empiezan a comercializar sistemas de gestión de bases de datos (DBMS), el lenguaje SQL (Structure Query Language) se estandariza como una herramienta para interactuar con el motor de bases de datos. Otro acontecimiento destacado en la década de los años 80 es el surgimiento de la inteligencia artificial (IA); con la aparición de este concepto muchos beneficios se obtuvieron, tales como tecnología de las máquinas de aprendizaje y algoritmos, que junto con las tecnologías de las bases de datos, capacidades de almacenamiento y computación paralela emerge la industria de minería de datos (datamining); primero como investigaciones académicas en donde sobresale por ser una colección de algoritmos de dominio público, y finalmente como productos robustos con fines comerciales. La industria ha madurado aceleradamente en los últimos 15 años para proveer ventajas reales a quien hacen uso de ella.

¿Qué son las bases de datos y como contribuyen a la inteligencia de datos?

Ya se ha mencionado que la materia prima para la minería de datos son los datos en general (de diversa procedencia). Las **Bases de datos** son estructuras lógicas organizadas en ficheros que se utilizan para almacenar datos de forma segura, consistente e íntegra de algún negocio, proceso o sistemas de información, y estos representan la fuente principal de datos para las actividades de análisis de datos.

El almacenamiento de grandes cantidades de datos no siempre garantiza el hallazgo de información útil, tendencias actuales se apoyan en la concepción de un "gobierno de datos" dentro de las organizaciones para identificar la significancia de su registro y gestión.

1.2.3. Algoritmos evolutivos y el poder de computo

Los 20 años pasados de nuestra economía ha sido una transición dentro de una era en que la información ha ido ganando posiciones como elemento valioso en

las organizaciones, de esta forma han llegado a ser las bases para tomar decisiones en muchas industrias. Las formas tradicionales de buscar información se han ido quedando en la obsolescencia, tanto por las capacidades de cómputo como por la decadencia de técnicas de exploración y búsqueda. Esto ha dado lugar a una nueva corriente de análisis de datos basados en la computación evolutiva, factores que han influido en esta tendencia son:

- Las organizaciones están continuamente recolectando grandes volúmenes de información sobre sus clientes, productos, procesos, ventas y muchas acciones adicionales.
- La propagación de métodos de captación de datos para contenidos de diferentes tipos.
- La continua evolución en el campo de máquinas de aprendizaje, con la contribución de nuevos algoritmos y estructuras de sistemas para optimizar el uso de hardware y software.
- Madurez en la importancia de la minería de datos como una fase previa para descubrir conocimiento en grandes volúmenes.

La **computación evolutiva** consiste en la artificialización de métodos de búsqueda basados en el comportamiento evolutivo natural de algunas especies del planeta.

En general, la idea común detrás de todas estas técnicas es la misma: dada una población de individuos, la presión ambiental causa la selección natural (supervivencia del más fuerte) y, por lo tanto, la calidad de la población va creciendo.

Como un proceso de optimización, se dice que, dada una función objetivo para maximizar, podemos crear aleatoriamente un conjunto de soluciones candidatas y usar la función objetivo como una medida de aptitud abstracta (cuanto más alta, mejor).

Para (Z. Li et al., 2020) “Las estrategias de evolución son populares debido al potencial de resolver problemas complejos de optimización de caja negra. A diferencia de los métodos basados en gradientes, las estrategias de evolución basadas en la población son más resistentes a la aspereza del paisaje, como el ruido, la no convexidad y la multimodalidad.” Los procesos evolutivos basados

en computadora también se pueden utilizar como solucionadores de problemas eficientes para tareas de optimización, manejo de restricciones, aprendizaje automático y modelado.

En problemas de optimización y búsqueda, se han utilizado muchos algoritmos de computación evolutiva para abordar los desafíos de las técnicas tradicionales en términos de reducir el tiempo y el número de patrones extraídos de conjuntos de datos a gran escala.

- *Algoritmos basados en evolución,* son métodos de búsqueda estocásticos basados en ideas evolutivas de la selección natural y la genética. Estos algoritmos utilizan operadores biológicos como el cruce, la mutación y la selección natural.
- *Inteligencia basada en enjambres,* los algoritmos de inteligencia de enjambres se inspiran en el comportamiento colectivo de enjambres como pájaros, peces, abejas y colonias de hormigas.
- *Algoritmos inspirados en principios de la física,* las metaheurísticas inspiradas en la física simulan los comportamientos físicos de la materia o siguen las leyes de la física.
- *Hibridación,* la hibridación de diferentes mecanismos de búsqueda es una solución que combina las ventajas de algunos algoritmos de la computación evolutiva para equilibrar la exploración y explotación.

1.3. Retos y aplicaciones

A cada momento se están desarrollando investigaciones que buscan aplicar las técnicas de minería de datos o el proceso de descubrimiento de conocimiento (KDD) para resolver un determinado problema concerniente a diversas áreas de la ciencia, de carácter social, psicológico, tecnológico, etc.

Una vista general de la aplicación de la minería de datos puede ser derivado de la naturaleza de las técnicas empleadas, así:

- Clasificación

- Estimación
- Pronóstico
- Asociación
- Segmentación o agrupación

La minería de datos puede ser aplicado a cualquier problema que involucre la presencia de datos, haciendo énfasis en conjuntos de datos de gran volumen y variados en sus tipos.

El éxito que han reflejado estas herramientas en las tomas de decisiones a nivel empresarial y tecnológico da lugar a que organizaciones y personas de diversas regiones del mundo tomen conciencia de la importancia de los datos.

Principalmente se pueden citar algunas áreas donde la minería de datos ha tenido mayor aportación (Gordan et al., 2022).

- Proveedores de servicios móviles
- Ventas y facturación
- Inteligencia artificial
- Comercio electrónico
- Ciencia e ingeniería
- Criminalística
- Investigación
- Farmacología
- Transporte
- Seguros

Algunos aportes específicos en el área de la salud son:

En la salud ha incrementado significativamente la dependencia de los datos, investigadores de la salud, médicos y farmacéuticas responden eficientemente a los problemas de salud actuales cuando tienen acceso a variedades de sistemas de información clínicos con grandes bases de datos (Zwolenski & Weatherill, 2014). Las bases de datos de los sistemas de información clínicos tienen grandes volúmenes de registros de historias clínicas de pacientes, diagnósticos médicos, información de monitoreo de pacientes, datos que

resultan bastante útiles en los sistemas de soporte a las decisiones clínicos (CDSS) para salvar vidas (Ara Shaikh et al., 2022).

Investigaciones aplicadas como enfermedades coronarias del corazón (CHD) que es un problema de salud bastante serio que causa muchas muertes. La inteligencia artificial (IA), la técnica de extraer información de una base de datos compleja utilizando sofisticados algoritmos informáticos, se ha incorporado al campo de la medicina. Las técnicas de IA han demostrado el potencial para acelerar la progresión del diagnóstico y el tratamiento de las enfermedades cardiovasculares (ECV), incluidas la insuficiencia cardíaca, la fibrilación auricular, la valvulopatía cardíaca, la miocardiopatía hipertrófica, la cardiopatía congénita, etc. (Sun et al., 2023)

Las metástasis cerebrales del cáncer de mama (BCBM) son las más mortales, con una supervivencia limitada en todas las metástasis a distancia del cáncer de mama. Investigadores usaron una base de datos SEER (2010-2019) para el análisis. Consistió en un análisis de regresión de COX para identificar los factores pronósticos de los pacientes con BCBM. A través de la validación cruzada, ellos construyeron modelos XGBoost para predecir la supervivencia en pacientes con BCBM. Concluyeron que los modelos XGBoost que tenían alta precisión y corrección, y fueron los modelos más precisos para predecir la supervivencia de los pacientes BCBM (Odhiambo et al., 2023).

En un estudio realizado en el estado regional de Afar (Etiopía) de febrero a junio de 2021. Se emplearon arboles de decisiones J48, red neuronal artificial, K-vecino más cercano, Máquina de vector de soporte (SVM), Regresión logística binaria, Random forest y Naïve Bayes utilizando datos secundarios de la revisión de registros de la base de datos médica. Se verificó la integridad de un total de 2239 conjuntos de datos de muestra diagnosticados con diabetes desde 2012 hasta el 22 de abril de 2020 (1523 con diabetes tipo 2 y 716 sin diabetes tipo 2) antes del análisis. Random forest, árbol de decisión J48 y k-vecino más cercano tuvieron un mejor rendimiento de clasificación y predicción para clasificar y predecir el estado de la enfermedad de la diabetes tipo 2 (Ebrahim & Derbew, 2023).

El uso de algoritmos para análisis de datos, creado sobre la base de la inteligencia artificial permite aumentar la probabilidad de embarazo en más de 2 veces. Así concluyeron los autores sobre la base del análisis del estado hormonal de una mujer y los datos de anamnesis con la aplicación de tecnologías de aprendizaje automático con la aplicación de redes neuronales artificiales, preparados sobre la base de miles de casos clínicos de embarazo exitoso, es posible predecir el más favorable, ciclos menstruales y fechas específicas de concepción (Buyanova et al., 2023).

Una propuesta de modelo, especialmente importante para situaciones en las que solo está disponible la información actual del paciente en lugar de tener múltiples puntos de datos de visitas regulares de niño sano espaciadas. Estos modelos predicen la obesidad utilizando información básica como el IMC al nacer, la edad gestacional, las medidas del IMC de las visitas de niño sano y el sexo. Los modelos pueden predecir la categoría de obesidad de un niño (normal, con sobrepeso u obeso) a los cinco años de edad con una precisión del 89 %, 77 % y 89% (Mondal et al., 2023).

En el área de marketing, también se cita un modelo eficiente para optimizar el sistema de venta de productos en una empresa farmacéutica utilizando métodos de agrupamiento y basado en indicadores y algoritmos de aprendizaje automático. El modelo estudiado utilizó índices RFM (Recency Frequency Monetary)-LRFM (Length Recency Frequency Monetary)-NLRFM (Number Length Recency Frequency Monetary) para aplicar algoritmos de agrupamiento de clientes. Además, el método de reglas de asociación se ha utilizado en este estudio para mostrar la relación entre los productos vendidos, analizar el carrito de compras de los clientes y ofrecer a los clientes en función de las reglas obtenidas. Finalmente, los resultados se revisan con algoritmos K-mean, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) y Optics. De acuerdo con los resultados obtenidos, el modelo propuesto proporcionará los mejores resultados utilizando el algoritmo K-means (Hamzehi & Hosseini, 2022).

1.4. Metodología para el análisis inteligente de datos

El término descubrimiento de conocimiento en bases de datos (KDD por sus siglas en ingles), se acuñó en 1989 para referirse al amplio proceso de encontrar conocimiento en los datos y enfatizar su aplicación en el desarrollo de minería de datos (U. Fayyad et al., 1996). A parte de KDD, son varias las metodologías, las que se han derivado para el desarrollo de proyectos de minería de datos tales como SEMMA (Sample, Explore, Modify, Model, Assess) SAS (2015), DMAMC (Definir, Medir, Analizar, Mejorar, Controlar), o CRISP-DM (Cross Industry Standard Process for Data Mining) (Wirth & Hipp, 2000).

Tabla 1
Cuadro comparativo metodologías minería de datos

KDD	SEMMA	CRISP-DM
PreKDD	----------	Comprensión del modelo
Selección	Muestreo	Comprensión de los datos
Preprocesamiento	Exploración	
Transformación	Modificación	Preparación de datos
Minería de datos	Modelado	Modelado
Interpretación	Evaluación	Evaluación
PostKDD	----------	Visualización

Nota: Autores (2023)

Un criterio de comparación entre KDD y SEMMA (ver Tabla 1), en principio se puede decir que son equivalentes, así: **Muestreo** se puede identificar con **Selección**; **Explorar** puede identificarse con **Preprocesamiento**; **Modificar** con **Transformación**; El **modelo** se puede identificar con **Minería de datos**, y **Evaluar** puede identificarse con **Interpretación**.

En el fondo se nota que las cinco fases del proceso SEMMA pueden verse como una implementación práctica de las cinco fases del proceso KDD. La metodología CRISP-DM incorpora pasos que deben preceder y seguir al proceso KDD (PreKDD, PostKDD), es decir: La fase de **Comprensión del modelo** se puede identificar con el desarrollo de una comprensión del dominio de la aplicación, el conocimiento previo relevante y los objetivos del usuario final; La fase de **Visualización** se puede identificar con la consolidación incorporando este conocimiento al sistema. En cuanto a las etapas restantes, podemos decir

que: La fase de **Comprensión de Datos** se puede identificar como la combinación de Selección y Preprocesamiento; La fase de **Preparación de Dato**s se puede identificar con Transformación; La fase de **Modelado** se puede identificar con DM; La fase de **Evaluación** se puede identificar con la de Interpretación. En la Tabla 1 se presenta un resumen de las correspondencias

Las tres metodologías citadas en resumen incluyen tres etapas muy bien definidas "preprocesado", "Minería de datos" y "Evaluación". Cada metodología utiliza su propia convención en el desarrollo de cada etapa.

1.5. El programa R-Project en la inteligencia de datos

R comenzó como un experimento, enfocado a usar las funciones de los implementadores de Lisp (Wikipedia contributors, 2023). El propósito era construir un pequeño banco de pruebas para ser usado en la construcción de un entorno estadístico. Al principio, se utiliza una sintaxis tipo S. Este es uno de los varios lenguajes de computación estadística que se diseñaron en los Laboratorios Bell y se formó por primera vez entre 1975 y 1976 (Bell et al., 1984). De tal forma que R tiene una alta similitud con este lenguaje de programación.

Varios acuerdos entre los programadores iniciales Robert Gentleman y Ross Ihaka, además de la persuasión de algunas comunidades estadísticas de la época hicieron que R esté disponible bajo la licencia GNU de la Fundación de Software Libre.

En general, R ahora ha superado sus orígenes vinculados a S, y ahora su desarrollo es un esfuerzo colaborativo. Los aficionados a la programación y contribuidores usan internet para intercambiar ideas y distribuir los resultados.

A medida que R crece en las diferentes comunidades, varios requerimientos fueron necesarios para acoger las sugerencias, recomendaciones y errores que los usuarios especificaban. Los mecanismos de interacción entre usuarios pronto llegaron a niveles de listas de correo. Las contribuciones de usuarios en lo referente a nuevas funcionalidades, también se hizo notable. Lo que dio la necesidad de estandarizar el proceso de distribución un archivo único, y la

implementación de varios sitios espejo, a parte del sitio maestro localizado en Austria.

1.5.1. Definiciones básicas

El programa R está dividido conceptualmente en dos partes:

- El sistema base de R, que se puede descargar desde CRAN
- Todo lo demás (librerías y extensiones)

La funcionalidad de R consta de paquetes modulares. El sistema base de R contiene el paquete básico que se requiere para su ejecución y la mayoría de las funciones fundamentales.

Tiene facilidad para obtener datos de diversas fuentes, desde archivos de texto basados en ASCII, hasta aquellos que provienen de programas comerciales como Excel, Matlab, SAS, SPSS, entre otros. También utiliza las rutinas SQL para obtener datos mediante consultas a bases de datos de diversos fabricantes.

Una característica que más llama la atención a los usuarios es su capacidad de gráficos impresionantes, quizá mejor que otras aplicaciones similares. El programa cuenta con una paquetería gráfica diversa, la librería ggplot2 es un referente para la construcción personalizada de visualizaciones para diversos tipos de gráficos.

Existen más de 5000 paquetes en CRAN, que han sido desarrollados por usuarios programadores alrededor de todo el mundo, además de aquellos paquetes que no se han distribuido formalmente.

A parte de ser un programa estadístico, R es un lenguaje de programación focalizado para trabajar con datos. Lo que hace posible el desarrollo de funciones, procedimientos y herramientas que integran interfaces gráficas. Algunos ejemplos son: Reclimdex, cliMTA-R, Rattle, etc.

Las comunidades de usuarios son muy activas, una lista extensa de foros, blogs, sitios web, portales web y muchas más, permite la resolución de preguntas y dudas, con ejemplos prácticos y amplias descripciones.

Las características de software libre, permite que usuarios con conocimiento avanzados puedan adentrase en el código fuente, para realizar adaptaciones

según requerimientos propios, redistribuir copias y liberar sus mejoras al público en general.

Constantemente crece en función de las necesidades para diversos tipos de usuario, actualmente se puede observar la capacidad de ejecutar código de Java en el entorno de R. Además, para los usuarios que construyen documentos tanto para la web como también en látex, R permite generar el formato de tales archivos con los datos desde sus estructuras.

R tiene una función de ayuda integrada similar a la función "*man"* de UNIX. Para obtener más información sobre cualquier función nombrada específica, por ejemplo "mean", el comando es

>help(mean)

>?mean

R es un lenguaje basado en expresiones algebraicas con una sintaxis muy simple. Al igual que un lenguaje de programación, es sensible a mayúsculas y minúsculas, por lo que "X" y "x" son diferentes y se referirían a variables diferentes.

Los nombres de identificadores o variables tienen especificaciones en cuanto al uso de caracteres, para evitar conflictos es recomendable seguir los estándares, que usan los lenguajes de programación convencionales.

Los comandos elementales consisten en expresiones o asignaciones. Por una parte, una expresión como un comando, se evalúa, y muestra su resultado en la consola. Por otra parte, una asignación también evalúa la expresión y pasa el valor a una variable, pero el resultado no se muestra automáticamente en la consola. Existe la posibilidad de definir varios comandos para un lote de ejecución, la separación está dado por el carácter ";", o puede utilizar el editor de scripts.

1.5.2.Objetos en memoria

Durante una sesión R puede crear varios objetos en memoria. Aunque al cerrar la sesión se libera toda la memoria ocupada, sin embargo, los hábitos de uso permanente pueden llegar a saturar la memoria, esto causa un bajo rendimiento

de R. El uso continuo de la aplicación puede dejar objetos innecesarios en la memoria, una forma correcta es su monitoreo constante para la liberación.

>objects()

>ls()

Los comandos indicados arriba se pueden usar para mostrar los nombres de los objetos que están actualmente almacenados dentro de R. La colección de objetos actualmente almacenados se llama espacio de trabajo.

Para remover los objetos de memoria se usa el comando siguiente:

>rm(x,y,datos1,aux,i)

Todos los objetos creados durante una sesión de R se pueden almacenar de forma permanente, en un archivo para su uso en futuras sesiones de R. Si indica que desea hacer esto, los objetos se escriben en un archivo llamado (.RData) en el directorio actual y las líneas de comando utilizadas en la sesión se guardan en un archivo llamado (.Rhistory).

Vectores y asignaciones

La estructura más simple es el vector numérico, que es un solo objeto que consta de una colección ordenada de números. Para configurar un vector llamado x, que consta de cinco números (0.4, 1.6, 1.1, 5.1 y 17.1), use el comando en R.

> x<-c(0.4,1.6,1.1,5.1,17.1)

La asignación también se puede realizar mediante la función assign().

>assign("x", c(0.4,1.6,1.1,5.1,17.1))

Si una operación se realiza con una variable de tipo vector en una expresión, esta operación involucra a cada elemento del arreglo. Así el comando:

> x+10

[1] 10.4 11.6 11.1 15.1 27.1

El comando “c” además de permitir los elementos uno a uno, también puede incluir objetos tipo vector, u otros comandos generadores de series.

```
> y=c(1:5,x,0.001)
```

Los operadores aritméticos elementales son +, -, *, / y ^ para elevar a una potencia. Además, todas las funciones aritméticas comunes están disponibles, tales como*: log, exp, sin, cos, tan, sqrt*, etc., todos tienen su significado habitual.

```
>   sum(x)/length(x) #Calcula la media de un vector
```

Generaciones de secuencias numéricas

R tiene una serie de facilidades para generar secuencias de números de uso común. Por ejemplo, 1:20 es el vector c(1, 2, …, 19, 20). El operador de dos puntos tiene alta prioridad dentro de una expresión, así, por ejemplo, 10*1:50 es el vector c(10, 20, …, 490, 500). También la generación puede ser en orden descendente cuando un valor mayor está primero 5:1.

seq() es una función más general para generar secuencias fácilmente. Tiene cinco argumentos, y solo algunos pueden especificarse en la llamada. Los dos primeros argumentos, cuando se especifican, definen el comienzo y el final de la secuencia, y si estos son los dos únicos argumentos dados, el resultado es el mismo que el operador de dos puntos. Eso es *seq(1,100)* es el mismo vector que 1:100. Cuando el tercer parámetro no es especificado el incremento es de 1, en el ejemplo de abajo el incremento se ha especificado en 10, y así se tiene:

```
> seq(1,5)
 [1] 1 2 3 4 5 #alternativa a 1:5
> seq(1,100,10) #10 es el incremento
   [1] 1 11 21 31 41 51 61 71 81 91
```

1.5.3.Operaciones básicas con datos

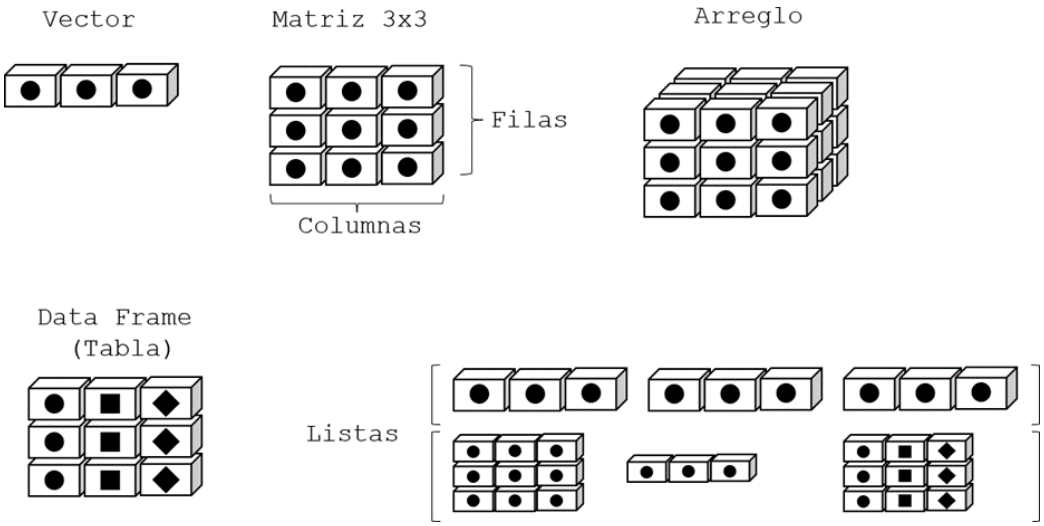
Una vez dado a conocer algunas de las operaciones básicas en R, se tiene un vistazo a algunas de las estructuras principales para almacenar los datos.

R es un lenguaje orientado a objetos y todas las estructuras de datos son objetos, además no proporciona acceso directo a la memoria, y se debe acceder a todos los datos a través de variables que se refieren a objetos.

Dado que la operación vectorizada es un aspecto importante de R. La estructura de datos básica es un **vector**, que es una secuencia de elementos de datos. Por lo tanto, un solo valor entero se trata como un vector entero de longitud uno. La estructura de datos más versátil es **list**, mientras que la más utilizada para el análisis de datos es el **data.frame.**

Figura 1

Representación lógica de las estructuras de datos de R



Nota: Falta fuente

data.frame es la estructura más utilizada para trabajar con datos, la flexibilidad para manipular diferentes tipos de datos lo hace atractivo.

La diferencia principal entre las diferentes estructuras de datos se encuentra en las dimensiones y los tipos de datos que se pueden almacenar. En la Figura 1 se ilustra mediante símbolos, la homogeneidad entre tipos de datos para las diferentes estructuras.

```
# ejemplos para crear vectores
v1=c(0.3,1.2,5.4,3.2,2.0)          # crea un vector numérico
v2=c("Juan","Julio","Mario")       # crea un vector de caracteres

# ejemplos para crear matrices
m1=matrix(1:16, nrow = 4, byrow = TRUE) #crea una matriz por fila
m2=matrix(c(3,4,2,8,2,1),ncol=2) #crea una matriz por columna
```

```
# ejemplos para crear arreglos
a1=array(1:6, c(3,4,2))# Crea un arreglo de 3 dimensiones a1=array(1:24, c(3,4,2))

# ejemplos para crear listas
ml1=list(list(1:6),list("x","y",1)) #Crea listas dentro de lista ids = c("x","y")
valores = c(2,3.5,1,9)
ml2=list(ids,valores) #Crea una lista de dos vectores

# ejemplos para crear data.frame
df1=data.frame(a=c("w","x","y"),b=13:15,c=T) # crea un data.frame de forma directa
m1=matrix(1:9,3,3)
df2=data.frame(m1) # crea un data.frame desde una matriz
```

R tiene gran cantidad de funciones asociadas a los tipos de datos, en lo que respecta a data.frame, varias operaciones de ajuste puede realizarse, así como: cambiar los nombres de variables, manipulación de filas y columnas (agregar, actualizar y eliminar), ordenamiento y filtración.

```
# Operaciones para agregar filas o columnas en data frames
> df1=cbind(df1,d=c(0,1,0)) #Añade la columna “d” al final
> df1=rbind(df1,c("z",16,F,1) #Añade una fila al final

# Modificando los nombres de columnas en data frames
> names(df1)=c("car","num","log","bin") #nuevos nombres de columna
```

>	df1			
	car	num	log	bin
1	w	13	TRUE	0
2	x	14	TRUE	1
3	y	15	TRUE	0
4	z	16	FALSE	1

Las expresiones comparativas y de búsqueda pueden especificarse a nivel de índices para las filas o columnas

```
# Ejemplo de algunas operaciones para filtrado de datos
```

```
> df1[df1$a %in% c("x","y"),]
> df1[c(2,3),]
> df1[which(df1$a=="x"|df1$a=="y" ),]
> df1[with(df1, a=="x" | a=="y"),]
> subset(df1, a=="x" | a=="y")
# Todas las instrucciones generan un mismo resultado
```

	a	b	c	d
2	X	14	TRUE	1
3	y	15	TRUE	0

*data.frame puede ser cargado desde archivos de diferentes formatos mediante las funciones de lectura en R, aunque existen herramientas visuales, lo más usual es mediante las variantes de "**read**". Sin embargo, es importante conocer el formato del archivo.*

```
archivo="D:\\datasets\\ejemplos\\empleados.csv"
dfArchivo=read.table(archivo,sep=";",dec=",",header=TRUE,encodin g = "UTF-8")
```

R cuenta con la función genérica read.table(), que puede leer cualquier tipo de archivo que contenga una tabla.

1.5.4.Uso de R para análisis inteligente de datos

Rattle (Williams, 2012) es un conjunto de herramientas de minería de datos gratuito y de código abierto escrito en el lenguaje estadístico R utilizando la interfaz gráfica de Gnome. Se ejecuta bajo GNU/Linux, Macintosh OS X y MS Windows.

Rattle es utilizado por aquellos que desean interactuar con R a través de una interfaz gráfica. El uso de la memoria en R representa un factor crítico de rendimiento, en CPU de 32 bits está limitado a conjuntos de datos relativamente pequeños (entre 50000 a 100000, según lo que esté haciendo). La implementación de R en servidores de CPU múltiple de 64 bits con 32 GB de memoria principal proporciona una plataforma poderosa para la minería de datos.

Instalación de Rattle

1) Instalar R-Project en la computadora (procure que sea la última versión), está disponible en la página https://cran.r-project.org/
2) Instale el paquete rattle

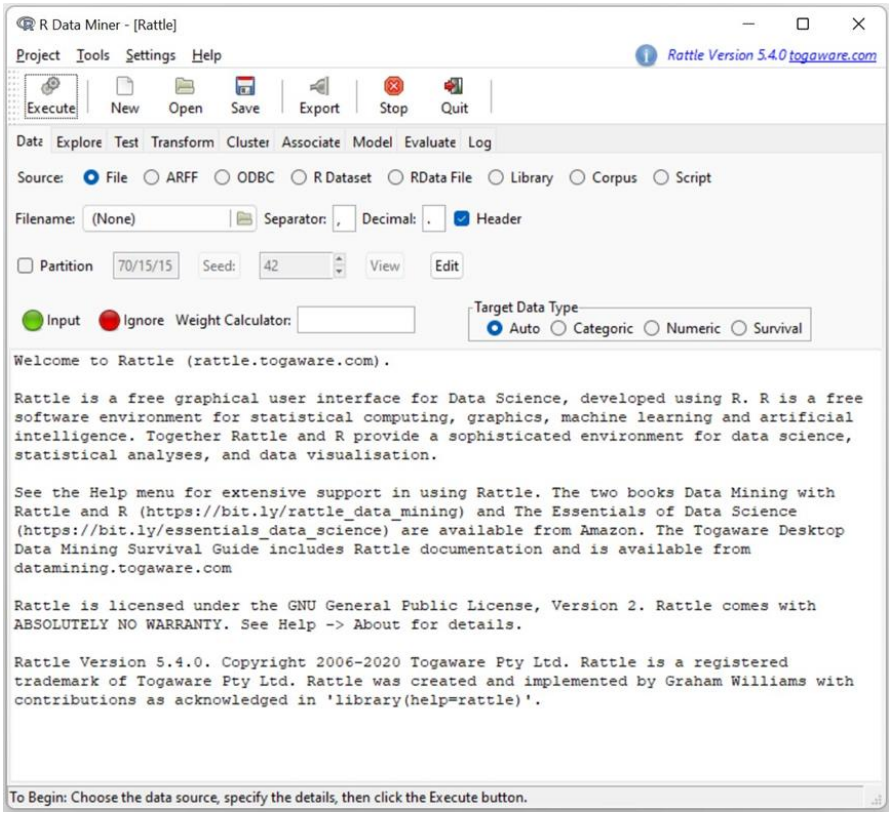
```
> install.packages("rattle")
> library(rattle)
> rattle()
```

Para la ejecución del módulo se requiere de la librería “RGtk2”, si no tiene instalado entonces ejecute la instrucción.

```
> install.packages("RGtk2")
```

Aunque de forma alternativa “rattle” al ejecutarse verificará y le pedirá la instalación del paquete. “RGtk2” es un paquete de R para crear interfaces gráficas de usuario (GUI) en R, es de código abierto y está escrito en lenguaje C

Figura 2
Interface principal de Rattle



Nota: Autores (2023)

Con Rattle (Figura 2) se tiene una interface gráfica intuitiva para utilizar R en procesos de minería de datos. Dispone de interfaces para la carga de datos desde diferentes fuentes de datos mediante la personalización de parámetros, resúmenes de datos, particionamiento de los datos para entrenamiento y prueba, transformación, algoritmos para reglas de asociación, agrupamiento y construcción de modelos (árboles de decisión, SVM, lineal y redes neuronales). Además, se incluye algunas pruebas estadísticas importantes para hipótesis con datos, y el uso de métricas para la evaluación de los modelos

Rattle tiene un registro histórico, donde se genera el código R correspondiente a las acciones del usuario en las interfaces gráficas.

02 CAPITULO

TÉCNICAS EN R PARA LA PREPARACIÓN DE LOS DATOS

Técnicas en R para la preparación de los datos

2.1. Introducción y objetivos

Comencemos relacionando a la información como un producto resultante de un proceso, así de forma análoga en un contexto industrial se puede asegurar que la calidad del producto depende de la calidad de la materia prima. Asimismo, la calidad de la información depende de la validez de los datos.

Los datos en el mundo real están en diferentes representaciones, debido a la naturaleza de su procedencia. Así, lo ideal sería encontrarse con datos estructurados que procedan de bases de datos u otros formatos basados en archivos digitales, en general que tengan una organización. Sin embargo, hoy en día existen variedades de mecanismos para la generación de datos tanto automáticos como manuales, que no siempre tienen una estructura definida.

El reto en la fase de preparación de datos es lograr que los datos tengan la calidad requerida para el análisis. Nótese, que es una etapa muy sensible que requiere mucha responsabilidad, debido a que los resultados posteriores dependerán de este paso.

Al finalizar este capítulo los lectores estarán en la capacidad de:

- Utilizar las estructuras de datos disponibles en R para operar con conjuntos de datos.
- Obtener resúmenes descriptivos con respecto a datos faltantes.
- Aplicar diferentes técnicas para imputación de datos faltantes.
- Aplicar procedimientos de discretización y/o normalización a los datos cuando éstos lo requieran.

2.2. Fuente de los datos

A menudo se necesitan diversidad de conjuntos de datos para realizar experimentaciones. Aunque dispongamos de un conjunto de datos propio, quizá algunos experimentos requieran de datos con variedades de características.

Acudir a los repositorios de acceso libre es una muy buena alternativa. Existen sitios en internet que tienen conjuntos de datos disponibles para ser descargados, muchos de ellos contienen información descriptiva, además de experimentos y publicaciones que se han realizado con ellos.

En general los datos pueden provenir de:

- Repositorios de acceso libre (contribuciones liberadas).
- Bases de datos transaccionales (siempre que se tenga el permiso).
- Mecanismos de captura de datos (sensores y alimentadores).

2.2.1. Repositorios de acceso libre

Hay muchos conjuntos de datos disponibles en la librería base y en diferentes paquetes de R. Las características de estos conjuntos de datos son diferentes, por ejemplo, algunos conjuntos de datos son series de tiempo, solo tienen columnas numéricas, tienen tanto números como factores o algunas incluyen columnas de caracteres.

Para obtener la lista de conjuntos de datos disponibles en la librería base de R, se puede usar:

```
> zdata()
```

También, se puede obtener un listado completo por paquete

```
> data(package = .packages(all.available=TRUE))
```

o los conjuntos de datos incluidos en una librería

```
> data(package = "nombre_librería")
```

Note que, para usar un conjunto de datos, se debe tener cargado la librería correspondiente, entonces acceder a los recursos o funciones incluidas.

```
> library(MASS)
> data(package="MASS")
```

```
Data sets in package 'MASS':

Aids2                           Australian AIDS Survival Data
Animals                         Brain and Body Weights for 28 Species
Boston                          Housing Values in Suburbs of Boston
    :
    :
    :
```

> dsBoston=Boston #Crea una copia del conjunto de datos

> head(dsBoston) #muestra las 6 primeras filas

```
     crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7
```

UCI Machine Learning Respository

El repositorio “UCI machine learning” (Dua & Graff, 2017) es una colección de conjuntos de datos que utilizan las comunidades de usuarios para el análisis empírico de los algoritmos de aprendizaje automático. La versión actual del sitio web fue diseñada en 2007 por Arthur Asunción y David Newman, y este proyecto es en colaboración con Rexa.info en la Universidad de Massachusetts Amherst.

Kaggle

Otro sitio donde se puede encontrar conjuntos de datos importantes es Kaggle, Kaggle es una plataforma gratuita que utiliza el concepto de “crowdsourcing”, es decir se basa en un modelo colaborativo que posee a disposición de los usuarios una serie de problemas para solucionar con temáticas como la ciencia de datos, el análisis predictivo y el aprendizaje máquina. Estos proyectos son patrocinados por compañías grandes que buscan soluciones de calidad a problemas de análisis de datos.

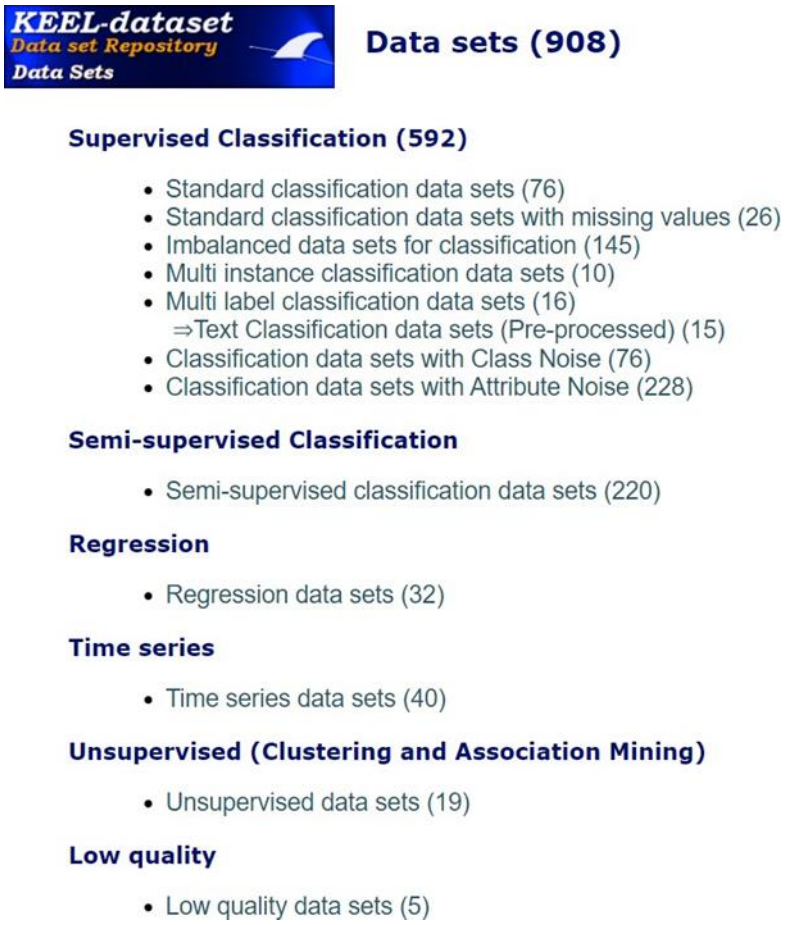
Repositorio de conjuntos de datos KEEL

Keel dataset repository (Alcalá-Fdez et al., 2011), KEEL (Knowledge Extraction based on Evolutionary Learning) es una suite de código abierto implementada en Java diseñada para utilizar diversas tareas de descubrimiento de conocimiento (Alcalá-Fdez et al., 2009).

Esta herramienta, además de permitir el diseño de experimentos basados en flujo de datos, con diferentes algoritmos de inteligencia computacional, incluye varios conjuntos de datos. La Figura 3 muestra los conjuntos de datos que dispone el sitio web de KEEL, clasificado por las técnicas de minería de datos generales.

Figura 3

Sección del repositorio de datos del sitio web KEEL



Nota: Autores (2023)

2.2.2. Bases de datos transaccionales

Las bases de datos transaccionales de una organización contienen muchos datos, resultado de los eventos diarios que registra. Sin embargo, es necesario contar con la autorización para su uso, algunas ventajas son:

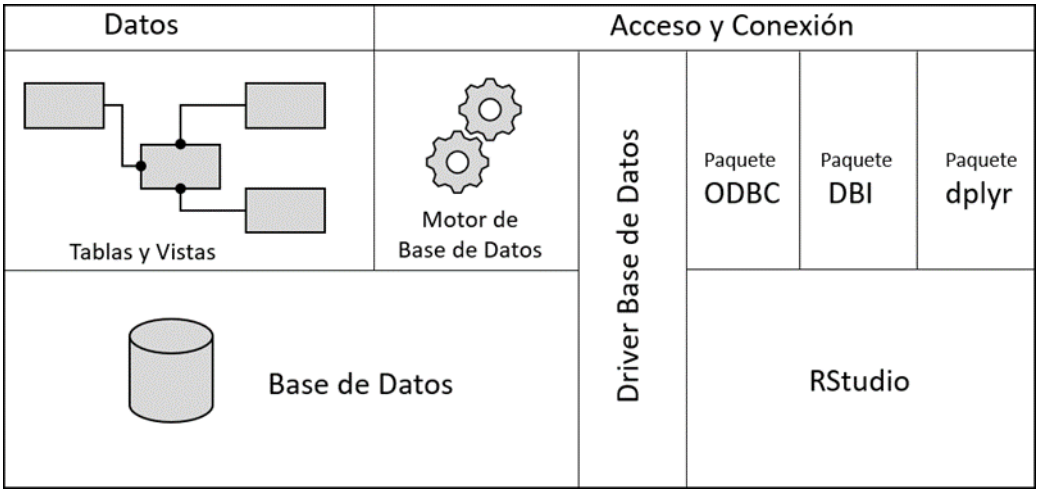
- Es poco probable que los datos hayan sido usados por otros investigadores

- Los datos son reales y es probable que presenten la realidad del comportamiento de todas las variables
- Se pueden generar reportes personalizados filtrando solamente lo requerido

Las bases de datos son creadas por personas con conocimientos de informática, de allí que se considere un buen diseño como proceso fundamental para la generación de reportes y datos. Muchas veces éstas no almacenan las variables que buscamos y entonces resultan poco útiles. Así algunas desventajas son:

- Problema de diseño estructural
- Es necesario el preprocesado
- Pueden tener mucho ruido

Figura 4
Esquema de interfaces entre R y RDBMS



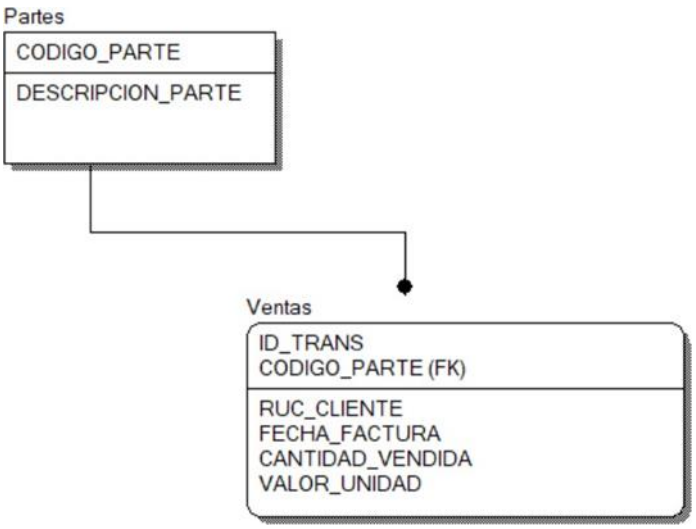
Nota: Autores (2023)

Desde un enfoque cliente/servidor, R puede tener acciones propias de una consola cliente para ejecutar tareas transaccionales sobre las bases de datos. Varios paquetes de R proveen funciones intuitivas que facilitan las tareas básicas, en especial las consultas que son las más útiles para la carga de datos externos.

A continuación, se muestra un ejemplo con una sección de la base de datos relacionada a ventas de partes de computadoras, la Figura 5 ilustra las dos tablas que son utilizadas en la demostración.

Figura 5

Sección de una base de datos ventas de partes computadoras



Nota: Autores (2023)

```
install.packages("RMySQL")
library(RMySQL)
# Crear un objeto de conexión para una Base de Datos MySQL.
# Conectaremos a la base de datos "bdcomputersparts"
mysqlConexion = dbConnect(MySQL(), user='javier',
password='javier1980', dbname='bdcomputersparts',
host='localhost')
#Listar las tablas disponibles en la base de datos
dbListTables(mysqlConexion)
```

Salida Código anterior:

```
[1] "partes" "ventas"
```

```
# Utilizando el objeto de conexión “mysqlConexion”
# Escribiremos una consulta SQL para obtener los 10 primeros
# registros de la table “partes”, y lo almacenamos en resultado
resultado = d
"s # Cargamos
la df_datos =
fe print (data.fr
```

Salida Código anterior:

	CODIGO_PARTE	DESCRIPCION_PARTE
1	00002507	ESFERO NEGRO | BIC
2	000L00123	PAPEL BOND A4 75GR | XEROX
3	001R00593	REP IBT BELT CLEANER ASSEMBLY 7132 | XEROX
4	001R00613	TRANSFER BEL CLEANER 7830 /XEROX
5	00216663	CUADERNO 100 HOJAS SENCILLO | JR

El conocimiento de lenguaje SQL y la estructura de la base de datos son requisitos esenciales para alcanzar una mayor personalización en la extracción de datos. Las líneas de código que siguen a continuación detallan un ejemplo para una consulta SQL a la base de datos de demostración.

Una vez almacenados los datos en las estructuras de R (por lo general un data frame), los datos pueden ser utilizados para los propósitos del usuario. Cabe indicar que la librería “RMySQL” no es la única, existen muchas interfaces más que tienen funciones como las citadas.

Una recomendación importante cuando se trabajan con cursores de conexión a servidores, en este caso a un motor de base de datos, es que se debe llevar control de las conexiones abiertas al servidor. Un cursor que se abre para ejecutar acciones en el servidor, deberá cerrarse luego de haber finalizado las transacciones, de esta forma se garantiza un rendimiento adecuado tanto del servidor como de la aplicación cliente.

Lista todas las conexiones al servidor MySQL dbListConnections(MySQL())

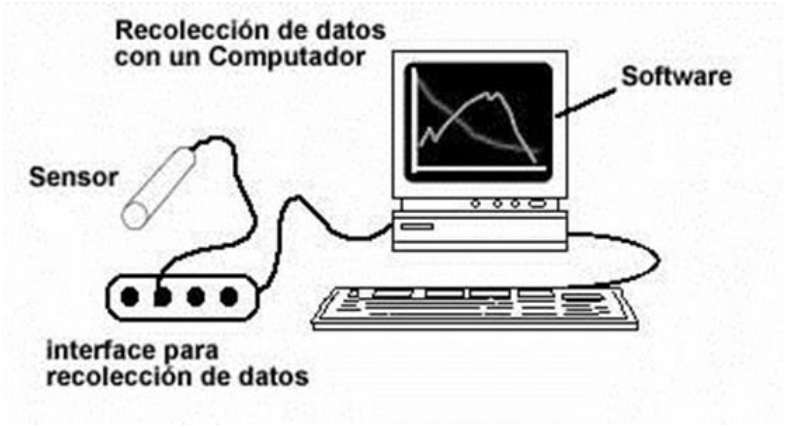
Desconecta el objeto mysqlConexion dbDisconnect(mysqlConexion)

2.2.3.Captura de datos desde dispositivos

Los datos también pueden provenir desde diversidad de dispositivos, estos tienen mecanismos basados en sensores ya sean de movimiento, luz, sonido, etc. Por lo general estos dispositivos contienen memorias temporales de almacenamiento o a su vez están conectados a través de alguna interface a una computadora maestro. Los datos se almacenan en formatos de archivos conocidos que pueden ser cargados desde R mediante las funciones básicas de lectura disponibles.

Figura 6

Representación básica de recolección de datos desde sensores



Nota: Autores (2023)

La Figura 6 ilustra una representación de recolección básica de datos, cuando estos provienen de dispositivos lectores como sensores. La combinación de elementos de hardware como de software permite construir mecanismos captura de datos personalizados. Los recursos provistos por los fabricantes son esenciales para una correcta sincronización y transferencia de datos con las aplicaciones y sistemas objetivos.

Como proyecto de código abierto, Arduino permite a las personas escribir código tipo C para programar un microcontrolador. También tiene un IDE, que viene con un monitor en serie. Como científico de datos, cabe la pregunta si podemos sacar los datos de esta caja y tal vez visualizarlos en R, de esta forma aprovechar más herramientas para generar mejores visualizaciones.

2.3. Selección de variables

Los modelos estadísticos son herramientas útiles aplicadas en muchos campos de investigación que tratan con datos empíricos. Relacionan una variable de resultado con una o varias de las denominadas variables independientes (VI), y cuantifican la fuerza de la asociación entre las VI y la variable de resultado. Al expresar estas asociaciones en cantidades simples e interpretables, los modelos estadísticos pueden proporcionar información sobre la forma en que cooperan varios factores para causar un resultado específico.

En las ciencias de la vida, muchos de estos mecanismos aún no se comprenden bien, pero a menudo es plausible suponer que son multifactoriales. Por lo tanto, la recopilación de datos empíricos y el análisis multivariable contribuyen de manera importante a la generación de conocimiento (Heinze et al., 2018).

Shmueli (2010) identificó tres propósitos principales de los modelos estadísticos. Dentro de las preguntas de investigación predictivas, los modelos predictivos (o pronósticos) tienen el objetivo de predecir con precisión un valor de resultado a partir de un conjunto de predictores. Los modelos explicativos se utilizan en la investigación etiológica y deben explicar las diferencias en los valores de los resultados por las diferencias en las variables explicativas. Finalmente, los modelos descriptivos deben “capturar la asociación entre variables dependientes e independientes”.

El propósito del modelado estadístico en un análisis particular tendrá un impacto en la selección de VI para un modelo.

¿Cómo se utilizan las variables de un conjunto de datos?

Dependiendo del tipo de variable resultante, los modelos más comunes pueden ser lineales/no lineales, logísticos y paramétricos. Estos modelos se hicieron populares en las ciencias de la vida porque los coeficientes de regresión (o sus transformaciones) son fácilmente interpretables. Las VI tienen el papel de “variables explicativas” en los modelos descriptivos y de “predictores” en los modelos predictivos.

En el caso de modelos lineales, los supuestos comunes son la linealidad, es decir, se cree que el valor del resultado esperado se modela mediante una combinación lineal de VIs, y la aditividad, es decir, se pueden agregar los efectos de las VIs. Existen varias extensiones de los modelos básicos de "predictores lineales" que pueden relajar la suposición de linealidad, como modelos polinómicos, splines, polinomios fraccionarios o modelos aditivos generalizados, pero no se considerarán aquí.

Un modelo lineal es definido por $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \varepsilon$, con Y como variable continua de resultado y $\varepsilon \sim N(0, \sigma^2)$ como el error distribuido de forma normal. En una configuración de varias variables independientes, la

interpretación de un coeficiente de regresión β_j en un modelo predictivo lineal es la del cambio esperado en el resultado. Así si X_j cambia por una unidad y la demás (X_1, ⋯ , X_{j-1}, X_{j+1}, ⋯ , X_k) se mantienen constantes, consecuentemente β_j mide el efecto condicional de X_j. Sin embargo, rara vez se puede suponer la existencia de un único modelo verdadero y, por lo tanto, de la especificación correcta del modelo. Esto implica que la interpretación de β_j cambia si el conjunto de VIs en el modelo cambia y Xj se correlaciona con otras VIs.

Un ejemplo tomado de (Heinze et al., 2018), aquí consideran un modelo que explique el porcentaje de grasa corporal por peso, altura y circunferencia abdominal, tres VIs altamente correlacionados.

El estudio se explica con más detalle en la Tabla 2, allí se muestra los coeficientes de regresión resultantes de cuatro modelos potenciales. Se observa que el coeficiente de peso cambia considerablemente en magnitud e incluso en su signo si se utilizan diferentes VIs para el ajuste; esto se debe a que el significado del coeficiente de peso es fundamentalmente diferente en los cuatro modelos. Además, el efecto de la altura puede considerarse irrelevante o altamente predictivo, dependiendo de si se ajustó o no la circunferencia del abdomen. La comparación del R2 ajustado de los modelos subraya el predominio de la circunferencia del abdomen. Por lo tanto, cualquier selección de variables aplicada en un modelo de predicción lineal con VI correlacionados siempre cambiará la interpretación de los efectos. Esto es de gran relevancia en modelos explicativos o descriptivos, donde hay interés en la interpretabilidad de los coeficientes de regresión.

Tabla 2

Cuatro modelos para estimar grasa corporal en porcentaje

	Coeficientes de regresión								
Modelo	**intercepto**		**Peso (kg)**		**Altura (cm)**		**Circunferencia abdominal**		
	Estimado	**Error Estándar**	**Estimado**	**Error Estándar**	**Estimado**	**Error Estándar**	**Estimado**	**Error Estándar**	**R2 ajustado**
1	-14.89	2.76	0.42	0.034					0.381
2	76.65	9.97	0.58	0.034	-0.58	0.062			0.543
3	-47.65	2.63	-0.29	0.047			0.979	0.056	0.722
4	-30.36	11.43	-0.22	0.068	-0.096	0.062	0.910	0.071	0.723

Nota: Extraído de Heinze et al. (2018)

2.3.1.Critério de significancia

Las pruebas de hipótesis son los criterios más populares utilizados para seleccionar variables en problemas prácticos de modelado. Sin pérdida de generalidad. Considere dos modelos $M1$: $\beta 0 + \beta 1X1 + \beta 2X2$ y $M2$: $\gamma 0 + \gamma 1X1$. La hipótesis nula que $\beta 2 = 0$ implica que $\beta 0 = \gamma 0$, así como $\beta 1 = \gamma 1$. Entonces tal hipótesis podría probarse comparando las probabilidades logarítmicas de $M1$y $M2$ (utilizando una prueba de razón de verosimilitud), lo que requiere el ajuste de los dos modelos.

Las pruebas de razón de verosimilitud (LRT) se utilizan para comparar la bondad de ajuste de dos modelos estadísticos. El LRT compara dos modelos anidados jerárquicamente para determinar si agregar complejidad a su modelo (es decir, agregar más parámetros) hace que su modelo sea significativamente más preciso.

En resumen, el LRT nos dice si es beneficioso agregar parámetros a nuestro modelo o si debemos quedarnos con nuestro modelo más simple.

En su forma más básica, las hipótesis para el LRT son:

H_0: debe usar el modelo anidado.

H_a: debe utilizar el modelo complejo.

Por lo tanto, si rechaza la H_0, puede concluir que el modelo complejo es significativamente más preciso que el modelo anidado y elegiría usar el modelo complejo. Si no puede rechazar la H_0, puede concluir que el modelo complejo NO es significativamente más preciso que el modelo anidado, por lo que elegiría usar el modelo anidado en su lugar.

La estadística de prueba para LRT sigue una distribución de chi-cuadrado con grados de libertad iguales a la diferencia en la dimensionalidad de sus modelos. La ecuación para la estadística de prueba se proporciona a continuación:

$$-2(loglikelihood(M1) - loglikelihood(M2))$$

Uso de R para pruebas de razón de verosimilitud

#descargue el paquete "lmtest" y llame a esa biblioteca para acceder a #la función lrtest().
library(lmtest)

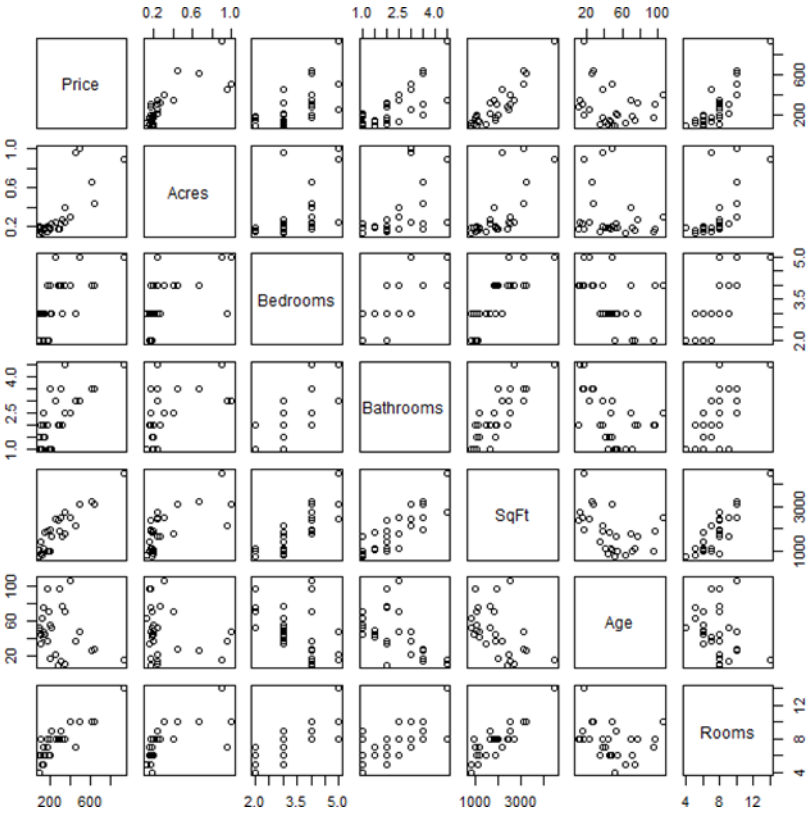
Comenzamos leyendo en nuestro conjunto de datos. Para este ejemplo, estamos leyendo datos sobre precio de las casas en función de una variedad de factores.

houseprices=read.csv("http://cknudson.com/data/Houseprices.csv", header=TRUE, sep=",")

Siempre es importante dar un vistazo a la correlación por pares en el conjunto de datos. Para ello, se usa el código a continuación para obtener esta vista y ver cómo las diferentes variables pueden correlacionarse entre sí.

plot(houseprices)

Figura 7
Matriz gráfica de correlación entre variables



Nota: Autores (2023)

Como se puede observar en la Figura 7, diferentes variables se correlacionan de con otras variables. Ahora, suponiendo que se quiere desarrollar un modelo para

los precios de las casas. ¿Qué variables ayudarían a crear el mejor modelo? ¿Se debería usar todas las variables o solo algunas de ellas?, Entonces se recurre al uso de LRT para obtener una ayuda.

Vamos a crear los modelos. Primero, es una buena idea obtener los nombres de las variables en nuestro conjunto de datos. También es una buena idea familiarizarse con el conjunto de datos. Puede usar diferentes funciones para visualizar los datos tanto de forma gráfica como tabular.

```
names(houseprices)
```

```
[1] "Price" "Acres" "Bedrooms" "Bathrooms" "SqFt" "Age"     "Rooms"
```

A continuación, se crean unos modelos utilizando variables que se piensa que pueden ser buenos predictores de los precios de las casas.

```
> simple <- glm(Price~Bedrooms,data=houseprices)
> complejo <- glm(Price~Bedrooms+Rooms,data=houseprices)
```

Finalmente, se hace el cálculo de la prueba estadística. Para ello, se halla las probabilidades logarítmicas de cada modelo y se remplaza en la fórmula.

```
-2 * [loglikelihood(simple)-loglikelihood(complejo)]
```

El estadístico de prueba sigue una distribución chi-cuadrado con grados de libertad iguales a la diferencia en el número de parámetros libres entre el modelo complejo y el modelo simple. Con esta información, se calcula el valor p, y si es menor que el nivel de significancia, se rechaza la hipótesis nula.

```
(A <- logLik(nested))
'log Lik.' -179.8936 (df=3)
(B <- logLik(complex))
'log Lik.' -168.893 (df=4)
> (teststat <- -2 * (as.numeric(A)-as.numeric(B)))
[1] 22.00128
#df = 4 - 3 = 1
> (p.val <- pchisq(teststat, df = 1, lower.tail = FALSE))
[1] 2.724687e-06
```

Un nivel de significación común para usar es 0.05. Bajo ese nivel de significación, se rechaza la hipótesis nula y se concluye que debe usar el modelo más complejo.

Veamos otro ejemplo. Observe los grados de libertad para nuestra estadística de prueba. Esto está codificado en el cálculo del valor p a continuación, así que no olvide cambiarlo del ejemplo anterior.

```
> simple <- glm(Price~Rooms+SqFt,data=houseprices)
>complejo <-glm(Price~Age+Rooms+Bathrooms+Bedrooms,data=houseprices)
> (A <- logLik(nested))
'log Lik.' -161.6652 (df=4)
> (B <- logLik(complex))
'log Lik.' -165.1707 (df=6)
> (teststat <- -2 * (as.numeric(A)-as.numeric(B)))
[1] -7.010964
#df = 6 - 4 = 2
> (p.val <- pchisq(teststat, df = 2, lower.tail = FALSE))
[1] 1
```

Con un nivel de significancia de 0.05, no se puede rechazar la hipótesis nula. Esto significa que en este caso debe usar el modelo simple en lugar del modelo complejo. Esto tiene sentido cuando observa las correlaciones por pares anteriores. Las variables “Rooms” y “SqFt” (número de cuartos y área de la casa) están claramente correlacionadas positivamente con los precios de las casas, mientras que otras variables muestran una correlación más débil.

2.3.2.Criterio de información

Si bien los criterios de significancia se aplican generalmente para incluir o excluir VI de un modelo, el enfoque de los criterios de información es seleccionar un modelo de un conjunto de modelos plausibles. Dado que incluir más VIs en un modelo aumentará ligeramente el ajuste aparente del modelo (expresado por medio de la probabilidad del modelo), se desarrollaron criterios de información para penalizar el ajuste aparente del modelo por la complejidad del modelo. En el trabajo desarrollado por (Akaike et al., 1973) se propuso aproximar la expectativa del logaritmo de verosimilitud con validación cruzada.

El criterio de información de Akaike (AIC) se formula de manera equivalente como: −2 log $L(x_{train}|\hat{\beta}_{train})$ + 2k (más pequeño es mejor), donde k es el número de parámetros del modelo. El valor predeterminado de k es 2, por lo que un modelo con solo una variable predictora tendrá un valor de k de 2+1 = 3. Mientras que log $L(x_{train}|\hat{\beta}_{train})$ es la verosimilitud del modelo, puede ser calculado directamente por un software estadístico.

AIC se puede utilizar para comparar dos modelos incluso si no están anidados jerárquicamente. También se puede emplear para seleccionar uno de varios modelos. Por ejemplo, a menudo se utiliza como criterio de selección en un procedimiento de selección del "mejor subconjunto" que evalúa todos los modelos (2k para k variables) resultantes de todas las combinaciones posibles de VIs.

Para calcular el AIC de varios modelos de regresión en R, podemos usar la función aictab() del paquete AICcmodavg.

El siguiente ejemplo muestra cómo usar esta función para calcular e interpretar el AIC para varios modelos de regresión en R.

Suponiendo que se quiere ajustar tres modelos de regresión lineal múltiple diferentes utilizando variables del conjunto de datos “mtcars” (disponible en R).

“mtcars” es un conjunto de datos que fue extraído de la revista estadounidense Motor Trend de 1974 y comprenden el consumo de combustible y 10 aspectos del diseño y rendimiento del automóvil para 32 automóviles (modelos 1973—74).

Estas son las variables predictoras que se usan en cada modelo:

Modelo 1: disp, hp, wt, qsec
Modelo 2: disp, qsec
Modelo 3: disp, wt

El siguiente código muestra cómo ajustar cada uno de estos modelos de regresión:

```
#tres modelos ajustados

modelo1=lm(mpg   ~ disp   +   hp + wt + qsec, data = mtcars)
```

```
modelo2=lm(mpg ~ disp + qsec, data = mtcars)
modelo3=lm(mpg ~ disp + wt, data = mtcars)
```

A continuación, se pone los modelos en una lista y se usa la función aictab()

para calcular el AIC de cada modelo:

```
library(AICcmodavg)
#lista con los modelos definidos
modelos <- list(modelo1, modelo2, modelo3)
#asignar nombres a los modelos
mod.names <- c('mod1', 'mod2', 'mod3')
#calcular AIC de cada modelo
aictab(cand.set = models, modnames = mod.names)
#resultado
Model selection based on AICc:
```

	K	AICc	Delta_AICc	AICcWt	Cum.Wt	LL
mod1	6	162.43	0.00	0.83	0.83	-73.53
mod3	4	165.65	3.22	0.17	1.00	-78.08
mod2	4	173.32	10.89	0.00	1.00	-81.92

A continuación, en la Tabla 3 se muestra una guía para interpretar el resultado.

Tabla 3

Significados Notación de resultados AIC

K	El número de parámetros en el modelo.
AICc	El valor AIC del modelo. La 'c' minúscula indica que el AIC se ha calculado a partir del AIC corregido para tamaños de muestra pequeños.
Delta_AICc	La diferencia entre el AIC del mejor modelo en comparación con el modelo actual que se compara.
AICcWt	La proporción del poder predictivo total que se puede encontrar en el modelo.
Cum.Wt	La suma acumulativa de los pesos AIC.
LL	El log-verosimilitud del modelo. Esto nos dice qué tan probable es el modelo, dados los datos que usamos.

Nota: Autores (2023)

El modelo con el valor AIC más bajo siempre aparece primero. A partir de la salida, podemos ver que el modelo 1 tiene el valor AIC más bajo y, por lo tanto, es el modelo que mejor se ajusta:

$$mpg = 27.33 + 0.03disp - 0.019hp - 4.61wt + 0.54qsec$$

2.4. Limpieza de datos

2.4.1. Definiciones

El uso del término “Limpieza de datos” es relativo, existen sinónimos utilizados en que se incluye este concepto. En general, la “limpieza de datos” surge debido a que los datos en el mundo real son “sucios” (haciendo referencia a inconsistencias, ruido, faltantes, duplicados e incoherencias en los datos).

Si no hay precisión en los datos, tampoco hay calidad en los resultados, las decisiones de calidad deben estar basados en datos de calidad.

En este manual, incluiremos tres procedimientos asociados a la limpieza de los datos.

1) Completamiento de los valores faltantes
2) Identificación de datos anómalos (outliers)
3) Corregir datos inconsistentes

Los datos no siempre están disponibles, muchas filas no tienen registrados valores para atributos, tales como los ingresos del cliente en datos de ventas.

La falta de valores se puede deber a:

- Mal funcionamiento de equipos.
- Inconsistencia con otros datos registrados y por lo tanto eliminados.
- Datos no ingresados debido a equivocaciones o malentendidos.
- Algunos datos pudieron no considerarse importantes al momento de ingresar datos.
- No se registró historial o cambios en los datos.

Los valores faltantes son un problema común en análisis estadístico. Se ha propuesto muchos métodos para el tratamiento de valores faltantes. Varios de estos métodos fueron desarrollados para el tratamiento de valores faltantes en encuestas por muestreo (Ge et al., 2023). Tratamiento de valores faltantes con regresión y tratamiento de datos faltantes en clasificación no supervisada (Jahangiri et al., 2023).

Estudios relacionados con clasificación supervisada:

- Chan and Dunn (1972) Imputación en LDA para problemas con dos clases.
- Dixon (1975) Imputación k-nn para lidiar con valores faltantes en clasificación supervisada.
- (Park et al., 2023) El problema de valores faltantes en aprendizaje supervisado usando redes neurales.

Impacto de valores faltantes:

- 1%, los datos faltantes son triviales.
- 1 a 5%, es manejable.
- a 20%, se requiere métodos sofisticados.
- 20% o más, el efecto perjudica las interpretaciones.

Mecanismos para valores faltantes

- Valores faltantes completamente al azar (MCAR)
- Valores faltantes al azar (MAR)
- Valores faltantes no al azar o no ignorables (NMAR)

2.4.2. Valores faltantes completamente al azar (MCAR)

La probabilidad que una instancia tenga un valor faltante para un atributo es la misma para todas las instancias. Es decir, esta probabilidad no depende ni de los valores observados ni de los valores faltantes. La mayoría de los valores faltantes no son MCAR.

Por ejemplo, supongamos que peso y edad son variables de interés en un estudio. Entonces los valores faltantes en el atributo peso son considerados

como MCAR, si la probabilidad que una persona de información acerca de su peso es la misma para todas las personas sin tomar en cuenta su peso y edad.

Este mecanismo es más adecuado para datos a ser usados en clasificación no supervisada.

2.4.3. Valores faltantes al azar (MAR)

La probabilidad que una instancia tenga un valor faltante en un atributo depende de los valores observados, como por ejemplo la clase a la cual pertenece la instancia, pero no depende de los valores faltantes. Este mecanismo es más adecuado para datos usados en clasificación supervisada.

2.4.4. Valores faltantes no al azar o no ignorables (NMAR)

La probabilidad de que una instancia tenga un valor faltante en un atributo depende de los valores faltantes en el conjunto de datos. Ocurre cuando las personas entrevistadas no quieren revelar algo muy personal acerca de ellas. El patrón de valores faltantes no es aleatorio. Este tipo de valores faltantes es el más difícil de tratar y es el que ocurre más frecuentemente.

2.4.5. Tratamiento de datos faltantes

Tomaremos como ejemplo el conjunto de datos “adult”, disponible en Repositorio UCI Machine Learning (Dua & Graff, 2017). El conjunto de datos original tiene 48842 observaciones, contiene variables continuas, ordinales y nominales (entrenamiento=32561, prueba= 16281). Cuando se eliminan las observaciones con valores faltantes quedan 45222 (entrenamiento=30162, prueba=15060), estos datos fueron donados por Ronny Kohavi y Barry Becker (1996). La Tabla 2.3 muestra los nombres de atributos para el conjunto de datos “adult”.

Tabla 4
Nombres de atributos del conjunto de datos Adult

age: variable continua.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: variable continua.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: variable continua.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. **occupation**: Tech-support, Craft-repair, Other-service, Sales, Exec- managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other- relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: variable continua. **capital-loss**: variable continua. **hours-per-week**: variable continua.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

income: >50K, <=50K

Nota: Autores (2023)

Figura 8

Muestra de una sección del conjunto de datos adult

	age	workclass	fnlwgt	education	educational.num	marital.status	occupation	relationship	...
1	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	...
2	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	...
3	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	...
4	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	...
5	18	?	103497	Some-college	10	Never-married	?	Own-child	...
6	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	...
7	29	?	227026	HS-grad	9	Never-married	?	Unmarried	...
8	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	...
9	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried	...
10	55	Private	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	...
11	65	Private	184454	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	...
12	36	Federal-gov	212465	Bachelors	13	Married-civ-spouse	Adm-clerical	Husband	...
13	26	Private	82091	HS-grad	9	Never-married	Adm-clerical	Not-in-family	...
14	58	?	299831	HS-grad	9	Married-civ-spouse	?	Husband	...
15	48	Private	279724	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	...
16	43	Private	346189	Masters	14	Married-civ-spouse	Exec-managerial	Husband	...
17	20	State-gov	444554	Some-college	10	Never-married	Other-service	Own-child	...
18	43	Private	128354	HS-grad	9	Married-civ-spouse	Adm-clerical	Wife	...
19	37	Private	60548	HS-grad	9	Widowed	Machine-op-inspct	Unmarried	...
20	40	Private	85019	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Nota: Autores (2023)

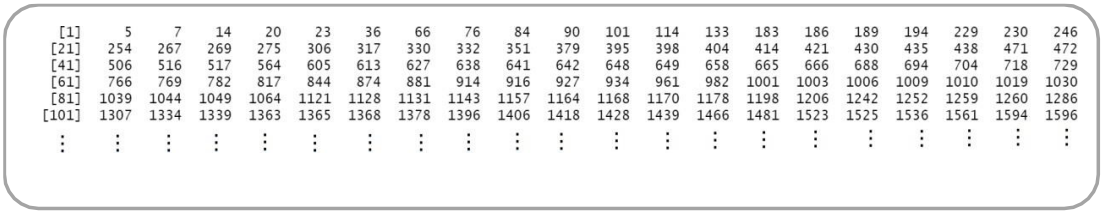
```
> ###Cálculos básicos para resumir datos faltantes
> pathAdult=".../data/adult_original.csv" #localización archivo
> datAdult = read.table(pathAdult, header = TRUE, sep =",",
                        dec=".",na.strings = "?")
> # importante utilizar el parámetro na.strings para identificar el
> # símbolo que corresponde al elemento faltante
```

Importante que en los datos cada elemento faltante quede etiquetado como <NA>, solo así R podrá reconocer como dato faltante, esto se logra con el parámetro ***na.strings,*** *tome en cuenta que el símbolo que representa al elemento faltante debe ser muy bien identificado. Para este caso ese simbolo es "?".*

```
> adultn=datAdult #Creamos una copia del conjunto de datos
> #Convertimos en numéricas todos los datos
> for(i in 1:15){
  if(!(is.numeric(datAdult[,i])))
  adultn[,i]= as.numeric(factor(datAdult[,i]))
  }
> #Para detectar las columnas con valores perdidos
> which(colSums(is.na(datAdult ))!=0)
```

workclass	occupation	native.country
2	7	14

```
> #Para detectar las filas con valores perdidos
> rmiss=which(rowSums(is.na(datAdult))!=0,arr.ind=T)
> rmiss
```



```
> #Para hallar el porcentaje de filas con valores faltantes
> length(rmiss)*100/dim(datAdult)[1]
 7.411654
> #Para hallar el porcentaje de valores faltantes por columna
> colmiss=c(2,7,14)
> pcolmiss= 100*colSums(is.na(datAdult[,colmiss]))/dim(datAdult)
> pcolmiss
```

workclass	occupation	native.country
5.730724	5.751198	1.754637

La disponibilidad de librerías que facilitan exclusivamente tareas de preprocesamiento en R no es conocido, existen funciones de preprocesado agregadas en librerías dispersas, tales como: discretización, normalización, imputación, etc. Sin embargo, no se identifica una que reúna todas las tareas, quizá se deba a la variedad de problemas que afrontan las tareas de preparación de los datos.

La librería "dprep" (Acuna & members of the CASTLE group at UPR-Mayaguez Puerto Rico, 2015) es un paquete que integra las funciones más utilizadas en esta etapa. Sin embargo, "dprep" está discontinuado a partir de la versión 3.x de R project. Para aprovechar sus funciones se recomienda la instalación de la versión de R y la agregación del paquete a partir de un archivo ".zip", que puede ser descargado de los registros históricos del programa R.

Para hacer los ejemplos que se muestran a continuación, se ha instalado la versión 2.15.3 de R Project y el paquete dprep v2.2.2

```
> #Resumen usando la función "imagmiss" de dprep
> imagmiss(adultn)
Report on missing values for:
Number of missing values overall: 6465
Percent of missing values overall: 0.9454685
Features with missing values (percent):
```

workclass	occupation	native.country
5.730724	5.751198	1.754637

```
Percent of features with missing values: 21.42857
Number of instances with missing values: 3620
Percent of instances with missing values: 7.411654
> #Eliminación de columnas y filas con valores faltantes
> adult.sup=clean(adultn,0.5,0.1,supr_adult)
```

Observe el segundo y tercer parámetro, representa el valor máximo de radio permitido para las columnas y las filas respectivamente. Con el valor de 0.5 para las columnas, no hay columna que sobrepase el 50% de valores perdidos para ser eliminadas. Con el valor 0.1 en las filas, en cambio si se encuentran

observaciones por encima de ese valor que serán eliminadas (tomando en cuenta el conjunto de datos trabajo, solo bastaría que una fila tenga al menos dos valores faltantes para ser incluida para eliminación).

	Variables	Percent.of.missing
1	workclass	5.73072355759387
2	occupation	5.7511977396503
3	native.country	1.75463740223578

Number of instances eliminated : 2799
Instance eliminated : 5 7 14 23 36 76 90 101 114 133 ...
Maximum number of values to be imputed: 821

2.4.6.Métodos para imputación de datos faltantes

Citamos a continuación tres formas con las que se pueden hacer frente a los escenarios de datos faltantes, cada uno tiene sus condiciones de aplicabilidad.

Eliminación de casos: Ignorar la fila que contiene datos faltantes. Usualmente es aplicado cuando el valor que falta es el de la clase (asumiendo que se está haciendo clasificación). No es efectiva cuando el porcentaje de valores faltantes por atributo varía considerablemente.

Estimación de parámetros: donde los procedimientos de Máxima Verosimilitud que usan variantes del algoritmo EM (Expectation Maximization) pueden manejar la estimación de parámetros en presencia de valores faltantes.

Técnicas de Imputación: donde los valores faltantes son reemplazados con valores estimados basados en la información disponible en el conjunto de datos.

En el contexto de clasificación supervisada, se usan cuatro métodos para el tratamiento de valores faltantes:

Eliminación de casos. Este método consiste en descartar todas las instancias (casos) con valores perdidos en por lo menos un atributo. Una variante de este método consiste en determinar el grado de valores faltantes en cada instancia y atributo, y eliminar las instancias y/o atributos con altos niveles de valores faltantes. Antes de eliminar cualquier atributo es necesario evaluar su relevancia en el análisis.

Imputación usando la media (MI). Reemplazar los valores faltantes de un atributo dado por la media de todos los valores conocidos de ese atributo en la clase a la que la instancia con el valor faltante pertenece.

```
> #Imputación usando media, función disponible en librería dprep
> adultn.meanImp=ce.mimp(adultn,c("mean"),1:13)
```

Imputación usando la mediana (MDI). Como la media se ve afectada por la presencia de outliers, parece natural usar la mediana en su lugar para asegurar robustez. En este caso los valores faltantes para un atributo dado son reemplazado por la mediana de todos los valores conocidos de ese atributo en la clase a la que la instancia con el valor faltante pertenece.

```
> #Imputación usando mediana, función disponible en librería dprep
> adultn.medImp= ce.mimp(adultn,c("median"),atr =1:13,nomatr=c(2,4:9,13))
```

Imputación con los K vecinos más cercanos

Dividir el conjunto de datos D en dos partes. Sea D_m el conjunto que contiene las instancias en las cuales falta por lo menos uno de los valores. Las demás instancias con información completa forman un conjunto llamado D_c

Para cada vector $x \in D_m$:

a) Dividir el vector en dos partes: una la de información observada y otra la de información faltante, x = [xo; xm]
b) Calcular la distancia entre xo y todos los vectores del conjunto Dc. Usar solo aquellos atributos en los vectores de Dc que están observados en el vector x.
c) Usar los K vectores más cercanos (K nearest neighbors) y considerar la moda como un estimado de los valores faltantes para los atributos nominales. Para atributos continuos, reemplazar el valor faltante por la media del atributo en la vecindad de los k vecinos más cercanos (knearest neighborhood)

El método no se podría aplicar si todas las filas de la matriz contienen al menos un valor faltante.

Usualmente, se toma k igual a 10. Pero el número de vecinos a usar es a lo sumo igual al número de filas completas de la matriz.

```
> #Imputación usando KNN, función disponible en librería dprep
> adult.knn = ec.knnimp(adultn,k=10)
> adult.knn = ec.knnimp(adultn,nomatr=c(2,4:9,13),k=10)
```

2.4.7.Comparación de tres métodos de imputación

Para este caso de estudio se usa “melanoma.csv”, es un conjunto de datos disponible en varios repositorios de acceso libre (tiene datos de 205 pacientes de una localidad de Dinamarca con melanoma maligno). No se detallará más especificaciones de los datos, debido a que se usa solo para el ejemplo de imputación.

```
#Preparar la tabla
melanoma = read.table(file, sep = ",", header = TRUE)
#reubicamos una variable de clase
melanoma <- melanoma[,c(1,2,3,8,5,6,7,4)]
#50 pruebas para identificar el mejor método
for(i in 1:50){
#Insertar datos faltantes de forma artificial
melanomaNA = melanoma
nMissVal = round(0.1 * nrow(melanoma), digits = 0)
idx = sample(nrow(melanoma), nMissVal, replace = TRUE)
melanomaNA[idx,7] = NA
#Imputacion por Media, Mediana y KNN
melanomaNA.mimp = ce.mimp(melanomaNA,c("mean"),7)
melanomaNA.mdimp = ce.mimp(melanomaNA,c("median"),7)
melanomaNA.knnimp = ec.knnimp(melanomaNA, k = 10)
#Datos imputados son separados en variables
origin = melanoma[idx,7]
mimp = melanomaNA.mimp[idx,7]
mdimp = melanomaNA.mdimp[idx,7]
knnimp = melanomaNA.knnimp[idx,7]
#Se calcula el error como medida de evaluación
```

```
error.mimp = sum((mimp - origin)^2)/nMissVal
error.mdimp = sum((mdimp - origin)^2)/nMissVal
error.knnimp = sum((knnimp - origin)^2)/nMissVal

# Se alamacena el historial de errores en la 50 pruebas
if(i==1){
dfError = data.frame(error.mimp, error.mdimp,error.knnimp)
} else {
dfError=rbind(dfError, c(error.mimp,error.mdimp,error.knnimp))
}
}
#Tabla de resultados con valores de error mínimo, máximo y
#promedio para cada técnica
dfResultado= data.frame("medida"=c("Minimo","Máximo","Promedio"),
                        "media"=c(min(dfError$error.mimp),
max(dfError$error.mimp),mean(dfError$error.mimp)),
                        "mediana"=c(min(dfError$error.mdimp),
max(dfError$error.mdimp),mean(dfError$error.mdimp)),
                        "knn"=c(min(dfError$error.knnimp),
max(dfError$error.knnimp),mean(dfError$error.knnimp)))
```

En la tabla que está a continuación se ve el resultado del experimento. El valor de error mínimo es para el método de la “mediana”, el máximo es para “knn” y el error promedio más bajo se tiene para el método de la “media”. Para este caso planteado se puede decir que el mejor método de imputación resultó el de la “media”, seguido de la “mediana” y finalmente “knn”.

	medida	media	mediana	knn
1	Mínimo	1.830497	0.86413	1.955962
2	Máximo	20.172215	20.93142	22.607260
3	Promedio	7.156096	7.55298	8.241050

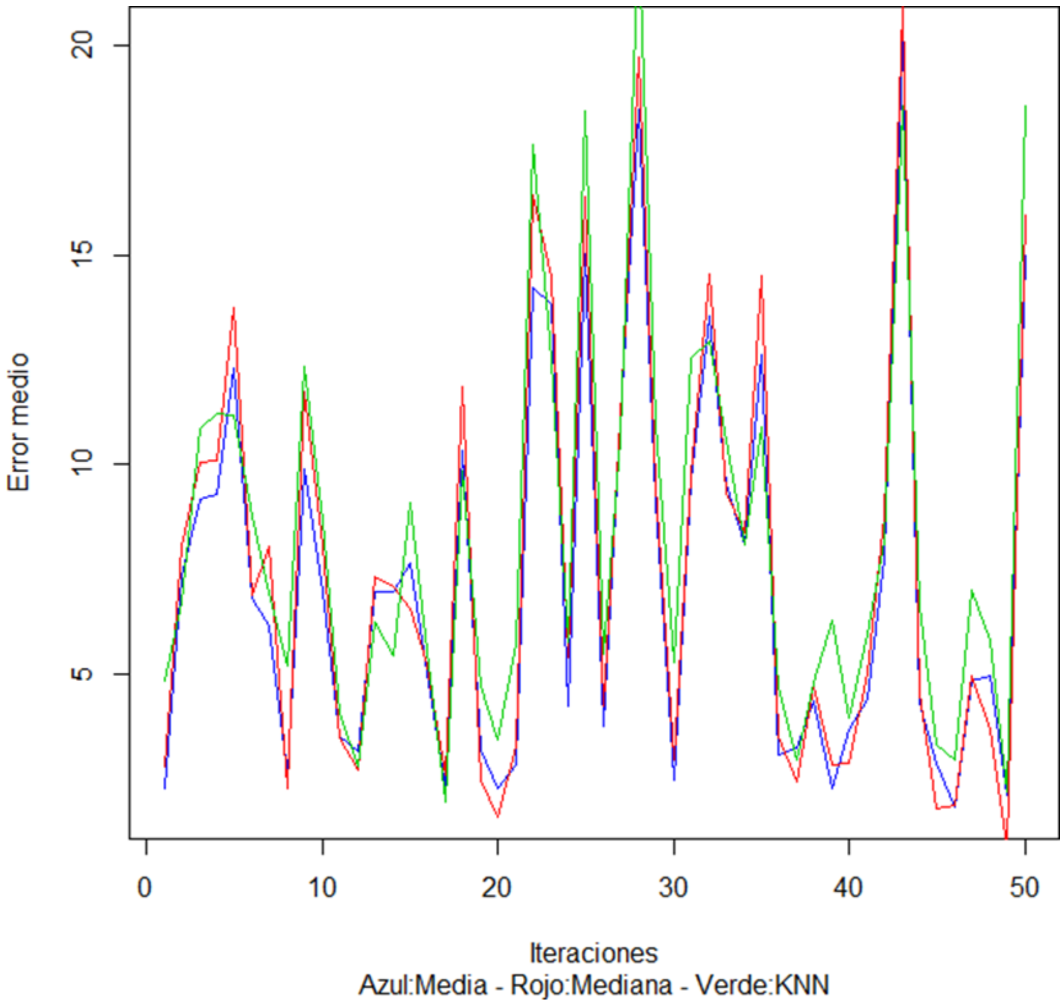
Un gráfico representativo de la comparación siempre es importante (ver Figura 9), un código sencillo para obtener un gráfico es el siguiente.

```
# Grafico comparativo de los tres métodos de imputación
```

plot(dfError$error.mimp,type= "l",col=4,main = "Gráfico comparativo de métodos de imputación",xlab = "Iteraciones", ylab = "Error medio", sub = "Azul:Media - Rojo:Mediana - Verde:KNN") *#Azul*
points(dfError$error.mdimp,type = "l",col=2) *#Rojo*
points(dfError$error.knnimp,type = "l",col=3) *#Verde*

Figura 9

Gráfico comparativo de tres métodos de imputación



Nota: Autores (2023)

2.4.8.Imputación por regresión lineal

La imputación puede tener un enfoque predictivo, de allí que se puedan aplicar algunas técnicas basadas en clasificación y regresión para la estimación de los valores faltantes.

A continuación, se indica un procedimiento basado en la regresión lineal.

El principio consiste en construir un modelo entrenado a partir de la parte completa de datos, luego utilizar el modelo para predecir los valores faltantes.

Pasos

1) Preparar el conjunto de datos (con valores faltantes).
2) Construir un modelo (y~x1 + x2 + ⋯ + xn) , donde y es la variable a predecir y X variables regresoras.
3) Construir una matriz con las filas de valores faltantes presentes.
4) Aplicar el modelo predictivo con la nueva matriz.
5) Completar el conjunto de datos original con los valores obtenidos en 4.

Para este ejemplo se ha seleccionado una muestra de 20 observaciones con 4 variables numéricas del conjunto de datos “pinguinos” disponible en la librería “datos”.

Tener el conjunto de datos con valores faltantes no implica que se deba cumplir el paso 1, pero para el ejemplo de demostración, los valores faltantes han sido insertados de forma artificial en la columna 2.

> dsPingui

A tibble: 20 x 4

	V1	V2	V3	V4
	<dbl>	<dbl>	<int>	<int>
1	36.6	18.4	184	3475
2	49	19.5	210	3950
3	42.5	NA	187	3350
4	43.4	14.4	218	4600
5	51.1	16.5	225	5250
6	48.6	NA	230	5800
7	40.6	19	199	4000
8	46.4	NA	216	4700
9	44	13.6	208	4350
10	35.3	18.9	187	3800
11	48.2	14.3	210	4600
12	34.6	NA	198	4400
13	49.2	18.2	195	4400
14	49.5	19	200	3800

15	49.8	17.3	198	3675
16	37.3	17.8	191	3350
17	40.1	18.9	188	4300
18	49	16.1	216	5550
19	40.5	17.9	187	3200
20	41.3	20.3	194	3550

Para utilizar la técnica de imputación basada en regresión es necesario preparar el conjunto de datos, de tal forma que solo una de las variables tenga valores faltantes. De forma alternativa, si hubiese más variables con datos faltantes se puede combinar para obtener otros modelos de regresión.

En el paso 2, el modelo es construido con las instancias que no tienen valores faltantes, así la variable dependiente será “V2” y las demás serán las regresoras.

```
#Obtiene las filas con datos faltantes
mv2=which(is.na(dsPingui[,2])!=0,arr.ind=F)
#Crea el modelo con las filas que están completas del dataset.
l1=lm(V2~., as.data.frame(dsPingui[-mv2,]))
```

El paso 3 prepara una matriz con las variables que no tiene valores faltantes, solo para las filas con datos NAs.

```
#crea un data frame con datos no NAs en filas que tienen faltantes.
a=as.data.frame(dsPingui[mv2,-2])
#Normalizamos los nombres de las variables
colnames(a)=c("V1","V3","V4")
```

La matriz “a” tiene los siguientes datos

```
> a
V1       V3    V4
1 42.5   187   3350
2 48.6   230   5800
3 46.4   216   4700
4 34.6   198   4400
```

Como paso 4 se aplica la predicción usando el modelo obtenido.

#Usamos la función predict para obtener los valores mediante el modelo

lmimp =predict(l1,a)

> lmimp

1	2	3	4
19.17760	14.00656	15.83362	16.65106

Finalmente, en el paso 5 completamos los valores en el conjunto de datos original.

> dsPingui [mv2,2]=lmimp

> dsPingui

A tibble: 20 x 4

	V1	V2	V3	V4
	<dbl>	<dbl>	<int>	<int>
1	36.6	18.4	184	3475
2	49	19.5	210	3950
3	42.5	19.2	187	3350
4	43.4	14.4	218	4600
5	51.1	16.5	225	5250
6	48.6	14.0	230	5800
7	40.6	19	199	4000
8	46.4	15.8	216	4700
9	44	13.6	208	4350
10	35.3	18.9	187	3800
11	48.2	14.3	210	4600
12	34.6	16.7	198	4400
13	49.2	18.2	195	4400
14	49.5	19	200	3800
15	49.8	17.3	198	3675
16	37.3	17.8	191	3350
17	40.1	18.9	188	4300
18	49	16.1	216	5550
19	40.5	17.9	187	3200

2.5. Discretización

2.5.1. Definiciones

La discretización es un procedimiento que permite transformar una variable continua en discreta. Las variables continuas en la base de datos pueden dificultar el aprendizaje en algunos algoritmos (Majid & Utomo, 2021).

Un proceso de discretización típico consta de cuatro pasos (X. Chen, 2020), que son:

1) clasificar los valores continuos de las características que se van a discretizar.
2) Evaluar los puntos de corte utilizados para la segmentación o la fusión de los intervalos adyacentes.
3) Dividir o fusionar los intervalos según algunos criterios.
4) Detenerse en un punto determinado.

Los algoritmos de discretización pueden dividirse principalmente en los siguientes tipos:

- Supervisados o no supervisados (En los algoritmos de discretización supervisada, no especifica el número de contenedores, y la discretización se ejecuta en función de los cálculos basados en la entropía y la pureza.);
- Globales o locales (los algoritmos globales agrupan los valores de cada característica en intervalos teniendo en cuenta otras características);
- Estáticos o dinámicos (los algoritmos estáticos discretizan cada característica en una iteración independientemente de otras características, mientras que los algoritmos dinámicos buscan todos los intervalos posibles para todas las características simultáneamente).

¿Por qué discretizar?

Uno de los motivos para discretizar un conjunto de valores es la necesidad de clasificar por intervalos los valores y así identificar clases con coherencia. Además de esto, puede buscarse maximizar la independencia entre los títulos de clases (Kotsiantis & Kanellopoulos, 2006).

Procedimiento general de la discretización

El procedimiento para discretizar generalmente empieza por el ordenamiento de los valores tomados en cuenta, seguido por la distinción de rangos, y finaliza con la clasificación de valores acorde a los rangos (Jahangiri et al., 2023).

De acuerdo con Liu et al, los pasos son 4, a los que han llamado *ordenamiento*, *evaluación*, *separación o unión*, y *detenimiento*. Aunque el punto de detenimiento consideramos que no es necesario incluirlo, esto puede ser un subproceso del penúltimo paso. A continuación, se detallan estos pasos.

Ordenamiento: Los valores continuos pueden ser ordenados de forma ascendente o descendente. El ordenamiento puede ser computacionalmente costoso, por lo que se debe elegir un algoritmo adecuado.

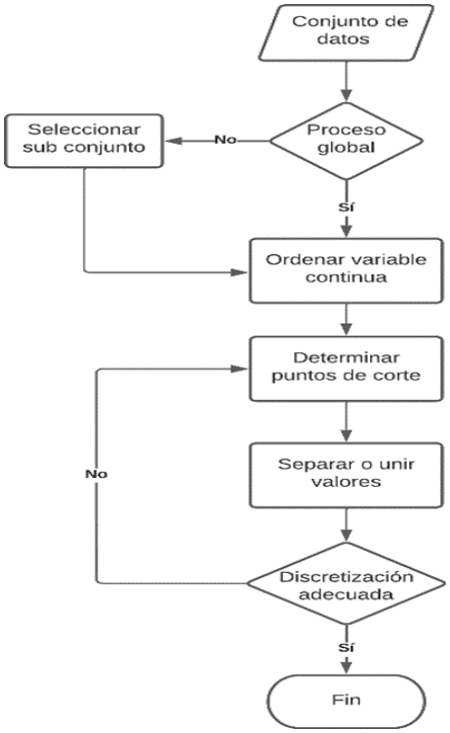
Evaluación: Este paso consisten en elegir los puntos de corte de los intervalos. Típicamente se determina la correlación de una división o fusión con la etiqueta de clase. Existen métodos para poder calcular los puntos de corte adecuados, por ejemplo, la entropía y alguna medida estadística.

Separación o unión: En este punto se establece si se debe “separar” o “juntar” los valores en los rangos ya establecidos. Para “separar” se toma un conjunto vacío de puntos de corte, y se van agregando nuevos puntos. Para “juntar” se tiene una lista completa de puntos de corte, para posteriormente eliminar algunos y así determinar los intervalos.

En la Figura 10 se observa un diagrama de flujo aproximado de cada paso.

Figura 10

Pasos para la discretización 1



Nota: Autores (2023)

2.5.2.Intervalos de igual frecuencia

El método de igual frecuencia (Hacibeyoglu & Ibrahim, 2018; Jiang et al., 2009) tiene como objetivo discretizar una variable continua en k intervalos con aproximadamente la misma cantidad de observaciones en cada intervalo. No es estrictamente necesario que cada intervalo tenga la misma cantidad de observaciones, pero sí que se aproximen lo mejor posible.

Para desarrollar este método es necesario ordenar los datos, ya sea de mayor a menor o viceversa. Posteriormente se requiere definir la cantidad de intervalos en los que se va a segmentar los valores, a esta variable la llamaremos k y se puede calcular con distintos métodos, para este estudio hemos seleccionado el método de Sturges.

$$k = log2(n + 1)$$

Luego se procede a discretizar los valores continuos. El proceso de discretización con igual frecuencias ignora la información de variables de clase.

A continuación, se muestra el proceso de discretización del conjunto de datos Melanoma. Para realizar la discretización se hará uso de 2 librerías y también se hará manualmente.

Se muestran las primeras filas de los datos antes de la discretización.

```
rownames(melanoma) <- NULL #No mostrar índices
kable(melanoma[1:10,])
```

X	time	status	sex	age	year	thickness	ulcer
1	10	3	1	76	1972	6.76	1
2	30	3	1	56	1968	0.65	0
3	35	2	1	41	1977	1.34	0
4	99	3	0	71	1968	2.90	0
5	185	1	1	52	1965	12.08	1
6	204	1	1	28	1971	4.84	1
7	210	1	1	77	1972	5.16	1
8	232	3	0	60	1974	3.22	1
9	232	1	1	49	1968	12.88	1
10	279	1	0	68	1971	7.41	1

Primero debemos preparar el conjunto de datos antes de empezar el proceso de discretización. Se procede a eliminar las variables irrelevantes, en este caso la variable *"x"*. Posteriormente ordenaremos las observaciones en función de la variable continua que se va a discretizar. La variable a discretizar será *"thickness".*

```
#Eliminar variable índice
melanoma = melanoma[,-1]
#Ordenar
melanoma = melanoma[order(melanoma$thickness),]
```

Vemos como quedan los datos con la variable continúa ordenada.

```
rownames(melanoma) <- NULL #No mostrar índices
kable(melanoma[1:5,])
```

time	status	sex	age	year	thickness	ulcer
2227	2	0	51	1971	0.10	0
355	3	0	64	1972	0.16	1
1512	2	0	77	1973	0.16	0
1542	2	0	65	1973	0.16	0
2431	2	0	49	1971	0.16	0

Ahora debemos definir el número de intervalos en los que se distribuirán los datos. En este caso utilizaremos el método de Sturges

```
k = round(log(length(melanoma[,1])+1, base = exp(2)), digits = 0)
## K intervalos =  3
```

A continuación, se hará uso de la librería **arules**. Para discretizar con esta librería se utiliza la función *discretize*, donde le indicaremos el método *method = "frequency"*, la cantidad de intervalos *breaks = k*, y podemos definir también las etiquetas con *labels = c("label 1", "label 2", …).*

```
#Paquete arules
library(arules)
#Discretizar
disc = discretize(melanoma$thickness, method = "frequency", breaks = k,
   labels = c("Low","Medium","High"))
#Agregamos la columna 'discrete' al dataset
melanoma$discrete = disc
#Número de elementos en cada intervalo
table(melanoma$discrete)
```

Como se puede observar, la librería *arules* cumple con el método de igual frecuencia. Intenta igualar la cantidad de elementos en cada intervalo, pero solo los aproxima, no se tiene la misma cantidad de observaciones en cada intervalo.

Esto es porque intenta que, cuando existe un cambio de intervalo, si existen dos valores iguales que están en diferentes intervalos, los agrupa dentro de uno solo. Los puntos de corte deberían ser en los índices 68 y 136. Veamos a continuación una tabla con los puntos en los que se cambia de un intervalo a otro.

	time	status	sex	age	year	thickness	ulcer	discrete
59	1654	2	0	67	1973	1.13	0	Low
60	2102	2	1	35	1972	1.13	0	Low
61	4479	2	0	19	1965	1.13	1	Low
62	1427	3	1	64	1972	1.29	0	Medium
63	1499	2	1	73	1973	1.29	0	Medium
64	1525	3	0	76	1970	1.29	1	Medium

	time	status	sex	age	Year	thickness	ulcer	discrete
133	5565	2	0	41	1962	2.90	0	Medium
134	1914	2	0	69	1972	3.06	0	Medium
135	2062	1	0	52	1965	3.06	0	Medium
136	232	3	0	60	1974	3.22	1	High
137	1435	1	1	27	1969	3.22	0	High
138	1812	2	1	44	1973	3.22	1	High

Librería Hmisc

Ahora probaremos la librería Hmisc. En este caso necesitamos indicarle la cantidad mínima de observaciones que deberá tener cada intervalo. Para esto se divide el total de observaciones del conjunto de datos para el número de intervalos deseados (k = 3).

```
#Librería Hmisc
library(Hmisc)

#Obtener número mínimo de observaciones
freq = floor(length(melanoma$thickness)/k) - 1

#Discretizar y mostrar resultados
table(cut2(melanoma$thickness, m = freq, g = 3))

##
## [0.10, 1.34) [1.34, 3.54) [3.54,17.42]
##           77           68           60
```

El problema de esta librería es que no tiene una función específica para el método de igual frecuencia, sino que se deben indicar varios parámetros para poder realizar la discretización. Sin embargo, sigue teniendo en cuenta los valores de la variable continua para realizar la discretización, lo que da un resultado similar a la librería anterior.

Método manual

Para este caso se elaboró un procedimiento manualmente, donde se definen las siguientes variables: *long = longitud del arreglo* freq = frecuencia o cantidad de observaciones por intervalo *labels = etiquetas para cada intervalo* min = índice de inicio para cada intervalo *top = índice de finalización para cada intervalo

Se realizan k iteraciones para ir completando un arreglo de valores discretos que corresponden a la clasificación o discretización de los valores de la variable continua. Las variables min y top se actualizan en cada iteración para determinar los límites del intervalo k.

```
#Método manual
long = length(melanoma$thickness) #longitud de la variable
freq = floor(long/k) #Frecuencia que se replicará en cada intervalo
labels = c("low", "medium", "high") #Etiquetas para cada intervalo (menor a mayor)
min = 1 #Primer índice del arreglo
#k iteraciones
for(i in 1:k){
  top = min + freq #Último índice de cada intervalo
  if(top > long){
        top = long     #top debe ser <= a la longitud del vector
  }
  melanoma$discrete[min:top] = labels[i] #Se discretiza con etiqueta
  min = top + 1 #Nuevo índice de inicio
}
```

Vamos a observar ahora algunas columnas del conjunto de datos ya discretizadas. Se ha colocado una columna adicional llamada *"discrete"* donde

se observa el resultado de la discretización, pero sin excluir la variable original *"thickness"*.

	time	status	sex	age	year	thickness	ulcer	discrete
1	2227	2	0	51	1971	0.10	0	low
5	2431	2	0	49	1971	0.16	0	low
36	1641	2	0	57	1973	0.81	0	low
45	3968	2	0	47	1967	0.81	0	low
74	3032	2	0	35	1969	1.29	0	medium
85	977	1	1	58	1967	1.62	1	medium
96	3476	2	0	60	1968	1.62	0	medium
103	2112	2	0	54	1972	1.94	1	medium
108	4124	2	0	30	1966	1.94	1	medium
114	1710	2	0	55	1973	2.26	0	medium
128	3278	2	1	54	1969	2.58	0	medium
133	5565	2	0	41	1962	2.90	0	medium
149	1779	2	1	55	1973	3.54	1	high
160	4103	2	0	52	1966	3.87	0	high
179	2165	2	1	62	1972	5.64	0	high
193	2542	2	1	67	1971	7.89	1	high

Al realizarlo de esta forma no se toma en cuenta los valores de la variable continua, por lo que vamos a encontrar algunos valores iguales en intervalos distintos, lo cual no sería adecuado.

	time	status	sex	age	year	thickness	ulcer	discrete
67	1652	2	0	58	1973	1.29	0	low
68	1678	2	0	59	1973	1.29	0	low
69	1804	2	0	48	1973	1.29	0	low
70	1839	2	0	40	1972	1.29	0	medium
71	1864	2	0	49	1972	1.29	0	medium
72	2521	2	0	45	1971	1.29	1	medium
73	2570	2	0	44	1970	1.29	0	medium

	time	status	sex	age	year	thickness	ulcer	discrete
136	232	3	0	60	1974	3.22	1	medium
137	1435	1	1	27	1969	3.22	0	medium
138	1812	2	1	44	1973	3.22	1	medium
139	2011	2	0	75	1972	3.22	1	high
140	2059	2	1	68	1972	3.22	1	high
141	2666	2	0	42	1970	3.22	1	high

2.5.3.Método 1R

Es un método de discretización supervisado que utiliza el binning. Después de clasificar los valores continuos, 1R divide el rango de valores continuos en un número de intervalos disjuntos y ajusta los límites en función de las etiquetas de clase asociadas a los valores continuos. Cada intervalo debe contener un mínimo de 6 instancias con la excepción del intervalo final que contendría las instancias restantes aún no agrupadas en ningún intervalo. El ajuste en el límite no permite terminar un intervalo si la siguiente instancia tiene la misma etiqueta de clase que la etiqueta de clase mayoritaria vista hasta entonces en este intervalo (H. Liu et al., 2002). Esto puede quedar más claro con el siguiente ejemplo sencillo.

Ejercicio práctico en R (1R con dprep)

Cargamos el conjunto de datos iris y se analiza las columnas que se pueden discretizar. La variable Species contiene datos de tipo categórico, y por lo tanto es descartada.

Una vez que tenemos los datos que necesitamos, procedemos a convertir todas las variables continuas a discretas

```
#-----Discretización usando el método 1r de Holte
#Cargamos los datos de iris
data(iris) iris
#convertimos los datos cualitativos en numéricos
iris[,5]=as.numeric(iris[,5])
iris
#discretizamos en un intervalo del 1 al 4
```

```
iris.1r=disc.1r(iris,1:4)
iris.1r[1:10,]
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1	15	1	1	1
2	1	8	1	1	1
3	1	12	1	1	1
4	1	11	1	1	1
5	1	15	1	1	1
6	1	15	1	1	1
7	1	15	1	1	1
8	1	15	1	1	1
9	1	7	1	1	1
10	1	11	1	1	1

La discretización no se puede realizar si existen valores faltantes o datos cualitativos, ya que este método es utilizado en bases de datos numéricos que requieran que una secuencia dividida en intervalos de los atributos continuos.

En el ejemplo mostrado, en la variable Petal.Width los valores fueron remplazados según la secuencia de los intervalos, siendo: 1 = [0.0 - 0.9], 2= [1.0 - 1.9] y 3=[2.0 – 2.9].

2.5.4.ChiMerge

En el problema de la discretización, se debe encontrar un compromiso entre la calidad de la información y la calidad estadística, así la calidad de la información se refiere a intervalos homogéneos en cuanto al atributo a predecir. Mientras que calidad estadística se enfoca en tamaño de muestra suficiente en cada intervalo para asegurar la generalización.

A manera de comparación, los criterios basados en la entropía se centran en la calidad de la información, mientras que los criterios de chi-cuadrado se centran en la calidad estadística.

En términos simples ChiMerge (Kerber, 1992) proporciona un método heurístico estadísticamente justificado para la discretización supervisada. Este algoritmo

comienza colocando cada valor real observado en su propio intervalo (inicialización), y continúa usando la prueba chi-cuadrado ($x2$) para determinar cuándo deben fusionarse los intervalos adyacentes. Este método prueba la hipótesis de que los dos intervalos adyacentes son estadísticamente independientes. Para ello, usa una medida empírica de la frecuencia esperada de las clases representadas en cada uno de los intervalos. El límite del proceso de fusión se controla mediante el uso de un umbral de $x2$ que indica el valor máximo de $x2$ que justifica la fusión de dos intervalos. Un dato importante es que en datos aleatorios se debe establecer un umbral muy alto para evitar crear demasiados intervalos

De forma simple, los pasos de este algoritmo se resumen a continuación:

1) Obtener el valor de χ2 para cada par de intervalos adyacentes.
2) Juntar el par de intervalos adyacentes que tengan el menor valor de χ2.
3) Repetir 1 y 2 hasta que los valores χ2 de todos los pares adyacentes excedan un valor umbral dado.
4) Umbral es determinado por el nivel de significancia y el grado de libertad (número de clases menos uno).

A continuación, se demuestra el comportamiento de chimerge con el conjunto de datos iris.

```
irisAux=iris
summary(irisAux)
##       Sepal.Length        Sepal.Width         Petal.Length        Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  Setosa     : 50
##  Versicolor : 50
##  Virginica  : 50
##
```

El paquete “discretization” contiene la función “chiM”.

```
library(discretization)
iris.disc<-chiM(irisAux, alpha = 0.1) # 10% en alfa
```

```
iris.disc$cutp #intervalos
##    [[1]]
##    [1] 4.85    4.95    5.45 5.75 6.25 7.05
##
##    [[2]]
##    [1] 2.45    2.85    2.95 3.35
##
##    [[3]]
##    [1] 2.45    4.75    5.15
##
##    [[4]]
##    [1] 0.80    1.35    1.75
```

Dentro de $cutp se encuentran los intervalos utilizados, así para la variable "Petal.Width" se tienen cuatro intervalos (arriba señala con el número [4] y corresponde a los intervalos {[0.1,0.80),[0.80,1.35),[1.35,1.75),[1.75,2.5]}

Dentro de $Disc.data se encuentran los datos discretizados.

```
head(iris.disc$Disc.data) #datos
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1            3           5            1           1  setosa
## 2            2           4            1           1  setosa
## 3            1           4            1           1  setosa
## 4            1           4            1           1  setosa
## 5            3           5            1           1  setosa
## 6            3           5            1           1  setosa
```

2.5.5.Entropía

La discretización basada en la entropía es un enfoque de división supervisado de arriba hacia abajo (U. M. Fayyad & Irani, 1993). Explora datos de distribución de clases en su cálculo y preservación de puntos de división (valores de datos para separación y rango de atributos)

Estos métodos de discretización utilizan medidas de entropía para evaluar los puntos de corte candidatos. Esto significa que un método basado en la entropía

utilizará la entropía de la información de clase de las particiones candidatas para seleccionar los límites para la discretización. La entropía de información de clase es una medida de pureza y mide la cantidad de información que se necesitaría para especificar a qué clase pertenece una instancia.

A continuación, se ilustra un ejemplo del procedimiento.

Sea S el siguiente conjunto de 12 pares (atributo valor, clase), S = {(2,F), (4,M), (10,F), (14,M), (20,M), (22,F), (28,M), (32, M), (36,F),(40,F),(42,F),(46,F)}.

Sea p_1 = 3/5 la fracción de pares con clase (F), y p_2 = 3/7 la fracción de pares con clase (M).

La entropía (o contenido de información) para S se define como:

$$Entropia(s) = -p_1 * log_2(p_1) - -p_2 * log_2(p_2)$$

Para ele ejemplo citado $Entropia(S)$ = 0.9798

Si la entropía es pequeña, entonces el conjunto es relativamente puro. El valor más pequeño posible es 0. Si la entropía es grande, entonces el conjunto está muy mezclado. El valor más grande posible es 1, el cual es obtenido cuando p1=p2=0.5.

Si el conjunto de muestras S es particionado en dos intervalos S_1 y S_2 usando el punto de corte T, la entropía después de particionar es:

$$Ent(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

donde | | denota cardinalidad. El punto de corte T se escoge de los puntos medios de los valores del atributo, así: {3, 7, 12, 17, 21, 25, 30, 34, 38, 41, 44}.

Por ejemplo, si T: valor de atributo=34.

S_1 = {(2,F), (4,M), (10,F), (14,M), (20,M), (22,F), (28,M), (32, M)}.

S_2 = {(36,F),(40,F),(42,F),(46,F)}.

$$Ent(S,T) = \frac{2}{3} * Ent(S_1) + \frac{1}{3} * Ent(S_2) = \frac{2}{3} * 0.9544 + \frac{1}{3} * 0 = 0.6362$$

Ganancia de información de la partición.

$$Gain(S, T) = Ent(S) - Ent(S, T)$$

$$Gain(S, T) = 0.9798 - 0.6362 = 0.3436$$

Igualmente, para T: va los de atributo = 12 se obtiene

$$Gain(S, T) = 0.9798 - 0.91829 = 0.0616$$

Por lo que v=34 es una mejor partición.

El objetivo de este algoritmo es encontrar la partición con la máxima ganancia de información. La ganancia máxima se obtiene cuando Ent(S,T) es mínima.

La mejor partición (es) se encuentran examinando todas las posibles particiones y seleccionando la óptima. El punto de corte que minimiza la función de entropía sobre todos los posibles puntos de corte se selecciona como una discretización binaria.

El proceso es aplicado recursivamente a particiones obtenidas hasta que se cumpla algún criterio de parada.

$$Ent(S) - Ent(S,T) > \delta$$

Donde,

$$\delta = \frac{\log(N-1)}{N} + \frac{\Delta(T,S)}{N}$$

$$\Delta(T, S) = log_2(3^c - 2) - [c \cdot Ent(S) - c_1 \cdot Ent(S_1) - c_2 \cdot Ent(S_2)]$$

Donde c es el número de clases en S, $c1$ es el número de clases en $S1$ y $c2$ es el número de clases en S_2. Esto es llamado el Principio de Longitud de Descripción Mínima (MDLP).

A continuación, se ilustra un ejemplo en R con la librería “discretización”, la función “mdlp” no tiene parámetros.

```
library(discretization)
irisAux=iris
iris.disc<-mdlp(irisAux)
iris.disc$cutp #intervalos generados

##  [[1]]
```

```
##   [1] 5.55        6.15
##
##   [[2]]
##   [1] 2.95        3.35
##
##   [[3]]
##   [1] 2.45        4.75
##
##   [[4]]
##   [1] 0.80        1.75
head(iris.disc$Disc.data) #datos discretos
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	1	3	1	1	setosa
2	1	2	1	1	setosa
3	1	2	1	1	setosa
4	1	2	1	1	setosa
5	1	3	1	1	setosa
6	1	3	1	1	setosa

Note que para la variable “Sepal.Length” se definen tres intervalos (arriba señala con el número [1] y corresponde a los intervalos {[4.30,5.55),[5.55,6.15),[6.15,7.90]}.

2.6. Normalización

2.6.1. Definiciones

La normalización de los datos consiste en limitar los valores propios de los datos preprocesados a un rango determinado, para eliminar los efectos adversos, como la lenta velocidad de convergencia del modelo causado por la gran diferencia entre los valores propios de las muestras (Shanwu et al., 2020).

Consiste en transformar los datos de forma que todos los predictores estén aproximadamente en la misma escala.

Es un paso de procesamiento previo de datos en el que ajustamos las escalas de las características para tener una escala de medida estándar.

La normalización es particularmente útil para algoritmos de clasificación que involucran redes neuronales, o mediciones de distancia como el vecino más cercano clasificación y agrupamiento. Si usa la red neuronal (algoritmo de retro propagación) para minería de clasificación, normalizando los valores de entrada para cada atributo, medido en las muestras de entrenamiento. Esto ayudará a acelerar la fase de aprendizaje. Para métodos basados en la distancia, la normalización ayuda a prevenir atributos con rangos inicialmente grandes (Al Shalabi et al., 2006).

2.6.2.Técnica Softmax

La función softmax (Goodfellow et al., 2016) transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

Características de la función softmax:

- Se utiliza cuando queremos tener una representación en forma de probabilidades.
- Esta acotada entre 0 y 1.
- Muy diferenciable.
- Se utiliza para normalizar tipo multiclase.

La función softmax es una función que convierte un vector de K valores reales en un vector de K valores reales que suman 1. Los valores de entrada pueden ser positivos, negativos, cero o mayores que uno, pero softmax los transforma en valores entre 0 y 1, para que puedan interpretarse como probabilidades. Si una de las entradas es pequeña o negativa, el softmax la convierte en una probabilidad pequeña, y si una entrada es grande, la convierte en una probabilidad grande, pero siempre permanecerá entre 0 y 1.

Muchas redes neuronales multicapa terminan en una penúltima capa que genera puntajes de valor real que no se escalan convenientemente y con los que puede ser difícil trabajar. Aquí, el softmax es muy útil porque convierte los puntajes en

una distribución de probabilidad normalizada, que puede mostrarse a un usuario o usarse como entrada para otros sistemas. Por esta razón, es habitual agregar una función softmax como capa final de la red neuronal.

Ecuación 1

$$V' = \frac{e^{\left(\frac{V - media(V)}{std(V)}\right)}}{\sum_{j=1}^{K} e^{z} j}$$

La ecuación (Ecuación 1) muestra la función softmax, donde V son elementos del vector de entrada y pueden tomar cualquier valor real. El término en la parte inferior de la fórmula es la expresión de normalización que asegura que todos los valores de salida de la función sumarán 1, constituyendo así una distribución de probabilidad válida. Mientras que K es el número de clases en un clasificador multiclase.

La función softmax y la función sigmoidal son similares. El softmax opera en un vector mientras que el sigmoide toma un escalar. *De hecho, la función sigmoidal es un caso especial de la función softmax para un clasificador con solo dos clases de entrada.*

Esto muestra que la función sigmoidal se vuelve equivalente a la función softmax cuando tenemos dos clases (Ver Ecuación 4)

En R la librería dprep contiene una función llamada **softmaxnorm**, la cual aplica la normalización softmax a una matriz o marco de datos, a continuación, se muestra un ejemplo básico.

Para el primer ejemplo se busca entender cuando aplicar la normalización de datos, como ya se ha mencionado con anterioridad, se aplica cuando hay diferentes variables con mayor rango que otros, es decir, una variable puede tener números entre 1 y 100, otra entre 1000 y 10000 y por último una entre 350 y 750, si se intenta graficar dichos valores se tendrá el siguiente resultado:

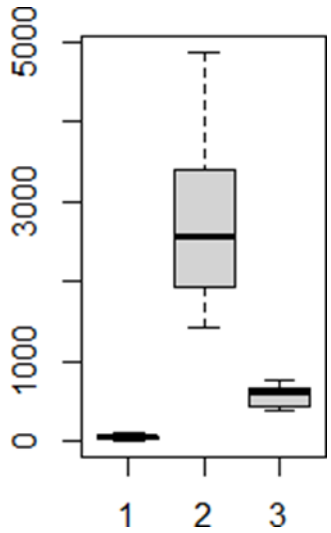
```
# Vector 1 con valores comprendidos entre 1 y 100
vec1 = c(sample(1:100, 10))

# Vector 2 con valores comprendidos entre 1000 y 5000
vec2 = c(sample(1000:5000, 10))
```

```
# Vector 3 con valores comprendidos entre 350 y 750
vec3 = c(sample(350:750, 10))

# Se crea una matriz cuyos valores son las tres variables antes declaradas
matrix.test = matrix(c(vec1, vec2, vec3), nrow=10, ncol=3)

# Se crea un gráfico para visualizar los datos
par(mfrow=c(1,2))
boxplot(matrix.test) #ver la Figura 11.
```

Como se puede observar la primera y tercera variable en el gráfico no se puede visualizar los valores en su totalidad ya que son muy pequeños a comparación con el otro conjunto de datos que comprende valores entre 1000 y 10000. Por tal motivo se debe aplicar un método de Normalización en este caso utilizaremos Softmax con la librería dprep.

Figura 11

Gráfico de cajas de diferencia de rangos técnica softmax



Nota: Autores (2023)

Dicho método busca acortar esos rangos disparejos que hay en los conjuntos de datos, este lo hace entre 0 y 1, para buscar tal efecto se debe invocar al método softmaxnorm pasándole como parámetro en conjunto de datos, a continuación, se muestra lo escrito:

```
# Se crea un variable de clase
matrix.test = cbind(matrix.test, rep(1, 10))
```

```
# Se utiliza la función softmaxnorm para normalizar los datos
matrix.softmax = softmaxnorm(matrix.test)
```

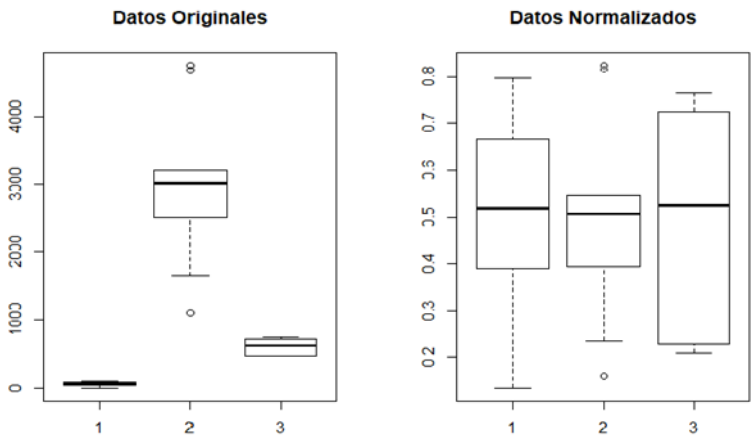
```
             [,1]      [,2]      [,3] [,4]
 [1,] 0.6965538 0.3600750 0.2179464    1
 [2,] 0.4218498 0.8640970 0.5769366    1
 [3,] 0.2817086 0.3177266 0.3380523    1
 [4,] 0.5874390 0.2548359 0.3603682    1
 [5,] 0.3987455 0.3736293 0.8207719    1
 [6,] 0.2447109 0.3292616 0.7993828    1
 [7,] 0.7065519 0.7259831 0.3007202    1
 [8,] 0.8102689 0.7816917 0.2804783    1
 [9,] 0.6863662 0.4015488 0.6742880    1
[10,] 0.1810775 0.4849673 0.5908560    1
```

Una vez obtenido los datos normalizados, se hace una comparación con los datos originales (Ver Figura 12), a continuación, se muestra el código:

```
par(mfrow=c(1,2))
boxplot(matrix.test[,1:3],main="Datos Originales")
boxplot(matrix.softmax[,1:3],main="Datos Normalizados")
```

Figura 12

Datos originales vs datos normalizados técnica softmax caso 1



Nota: Autores (2023)

Como se puede visualizar ahora los datos normalizados se encuentra en el mismo rango que define la función softmax, es decir entre 0 y 1, a comparación de los datos originales se ve más legible.

Ejercicio con datos reales, del conjunto de datos Melanoma

En este caso se hará lo mismo del anterior ejercicio, solo que esta vez los datos vienen de un archivo Excel, para cargar dichos datos se ejecuta lo siguiente:

```
# Se selecciona el archivo
fname = file.choose()

# Se carga el archivo con los datos
melanoma = read.csv(fname)
```

Para ese conjunto de datos se debe seleccionar variables que contengan datos de diferentes rangos, en este caso se selecciona la variable 1, 4, 6 y 7.

```
# Se crea una matriz con datos de diferentes rangos
melanoma.data = matrix(c(
  melanoma[, 1], melanoma[, 4], melanoma[, 6], melanoma[, 7]),
  nrow=nrow(melanoma),
  ncol=4 )
```

Se toma la última variable con clase y luego se procede ejecutar la función softmaxnorm

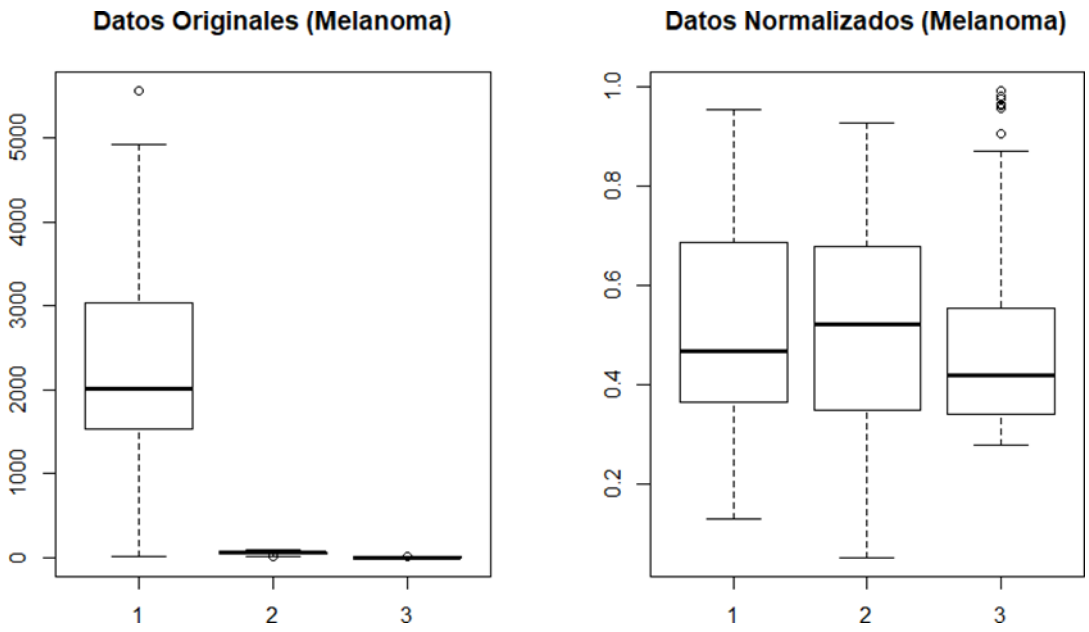
```
# Se utiliza la función softmaxnorm para normalizar los datos
melanoma.soft = softmaxnorm(melanoma.data
```

Por último, como en el anterior ejercicio se comparan los valores reales con los normalizados:

```
# Se crea un gráfico con los datos reales y los datos normalizados
par(mfrow=c(1,2))
boxplot(melanoma.data[,1:3],main="Datos Originales (Melanoma)")
boxplot(melanoma.soft[,1:3],main="Datos Normalizados (Melanoma)")
```

Figura 13

Datos originales vs datos normalizados técnica softmax caso 2



Nota: Autores (2023)

2.6.3.Técnica Z-SCORE

Los valores del vector V son normalizados en base a la media y desviación estándar (K.-P. Li & Porter, 1988).

Ecuación 2

$$V' = \frac{media(V)}{std(V)}$$

Este método trabaja bien en los casos en que no se conoce el máximo y mínimo de los datos de entrada o cuando existen outliers que tienen un gran efecto en el rango de los datos.

```
#Cargamos la librería dprep
library(dprep)
#Guardamos en irisTemp y preparamos los datos
irisTemp = iris
irisTemp.znorm =znorm(irisTemp) #uso de la función znorm
summary(irisTemp[,1:2])
summary(irisTemp.znorm[,1:2])
```

A continuación, se muestra el resumen de dos de las variables del conjunto de datos original y normalizado.

```
> summary(irisTemp[,1:2])
  Sepal.Length    Sepal.Width
 Min.   :4.300   Min.   :2.000
 1st Qu.:5.100   1st Qu.:2.800
 Median :5.800   Median :3.000
 Mean   :5.843   Mean   :3.057
 3rd Qu.:6.400   3rd Qu.:3.300
 Max.   :7.900   Max.   :4.000
```

```
> summary(irisTemp.znorm[,1:2])
  Sepal.Length      Sepal.Width
 Min.   :-1.86378   Min.   :-2.4258
 1st Qu.:-0.89767   1st Qu.:-0.5904
 Median :-0.05233   Median :-0.1315
 Mean   :0.00000    Mean   :0.0000
 3rd Qu.:0.67225    3rd Qu.:0.5567
 Max.   :2.48370    Max.   :3.0805
```

```
# Se crea un gráfico con los datos reales y los datos normalizados
par(mfrow=c(1,2))
boxplot(irisTemp[,1:3],main="Datos Originales (iris)")
boxplot(irisTemp.znorm[,1:3],main="Datos Normalizados (Iris)")
```

En la Figura 14 se observa la diferencia de rangos en los datos normalizados, se muestra solo las tres primeras variables del conjunto de datos “iris”

Figura 14

Datos originales vs datos normalizados técnica z-score

Nota: Autores (2023)

2.6.4.Otras técnicas de normalización

Normalización Min-Max. Este método realiza una transformación lineal de los datos originales V en el intervalo especificado [nuevo_min,nuevo_max]

Ecuación 3

$$V' = \frac{(V - min) * (nuevo\_max - nuevo\_min)}{(max - min) + nuevo\_min}$$

La ventaja de este método es que preserva exactamente todas las relaciones entre los datos. No introduce ningún potencial sesgo en los datos. La desventaja es que se encontrará un error "fuera del límite" si un futuro ingreso de datos cae fuera del rango original.

```
#Cargamos la librería dprep
library(dprep)
irisTemp.mmnorm =mmnorm(irisTemp, 0,10) #uso de la función mmnorm
summary(irisTemp[,1])
summary(irisTemp.mmnorm[,1])
```

Note en el cuadro de resultados la diferencia de rangos de la variable de ejemplo que se ha tomado, el nuevo rango se identifica en los valores Min y Max.

```
> summary(irisTemp[,1])
Min.   1st Qu.  Median  Mean   3rd Qu.  Max.
4.300  5.100    5.800   5.843  6.400    7.900
> summary(irisTemp.mmnorm[,1])
Min.   1st Qu.  Median  Mean   3rd Qu.  Max.
0.000  2.222    4.167   4.287  5.833    10.000
```

Normalización por escalamiento decimal. Es una forma sencilla de reducir los valores absolutos de un atributo numérico. Se normaliza moviendo el punto decimal de los valores de los datos.

Para normalizar los datos mediante esta técnica, se divide cada valor de los datos por el valor absoluto máximo de los datos. El valor de los datos, (V), de los datos se normaliza a (V') utilizando la fórmula a continuación:

Ecuación 4

$$V' = \frac{V}{10^j}$$

Donde j es el número entero más pequeño tal que el nuevo – max(|V'|) < 1, 1. Sólo es útil cuando los valores de los atributos son mayores que 1 en valor absoluto.

```
#Cargamos la librería dprep
library(dprep)
irisTemp. decscale=decscale(irisTemp) #uso de la función decscale
summary(irisTemp[,1])
summary(irisTemp.decscale[,1])
```

Observe en el cuadro de resultados como el punto decimal se ha movido hacia la izquierda para establecer los nuevos rangos.

```
> summary(irisTemp[,1])
Min.    1st Qu.  Median   Mean    3rd Qu.  Max.
4.300   5.100    5.800    5.843   6.400    7.900
> summary(irisTemp.decscale[,1])
Min.    1st Qu.  Median   Mean    3rd Qu.  Max.
0.4300  0.5100   0.5800   0.5843  0.6400   0.7900
```

Normalización sigmoidal. Si se busca una forma de reducir la influencia de valores anómalos en los datos sin tener que removerlos, la normalización sigmoidal es una opción. Los datos tienen una transformación no lineal mediante uso de una función sigmoidal, ya sea una función logística sigmoidal o tangente hiperbólica. La media y la desviación estándar se calculan para cada variable y se utilizan en la transformación dada en la ecuación (Ecuación 5). Con un sigmoidal logístico se pone los datos normalizados en un rango de 0 a 1. La ecuación (Ecuación 6) muestra la normalización con una tangente hiperbólica; esta pone los datos normalizados en un rango de -1 a 1.

Ecuación 5

$$V' = \frac{1}{1 + e^{-\left(\frac{V - media(V)}{std(v)}\right)}}$$

Ecuación 6

$$V' = \frac{1 - e^{-\left(\frac{V - media(V)}{std(V)}\right)}}{1 + e^{-\left(\frac{V - media(V)}{std(v)}\right)}}$$

A continuación, un ejemplo con la función que trae la librería “dprep”. Esta implementación corresponde a una tangente hiperbólica, por eso en el cuadro de respuesta se puede notar que el rango estaría entre [-1,1].

```
#Cargamos la librería dprep
library(dprep)
irisTemp.signorm=signorm(irisTemp) #uso de la función signorm
summary(irisTemp[,1])
summary(irisTemp.signorm[,1])
```

```
> summary(irisTemp[,1])
Min.    1st Qu.  Median   Mean     3rd Qu.  Max.
4.300   5.100    5.800    5.843    6.400    7.900
> summary(irisTemp.decscale[,1])
Min.      1st Qu.    Median     Mean       3rd Qu.   Max.
-0.731500 -0.420900  -0.026160  -0.007283  0.324000  0.846000
```

03 CAPITULO

TÉCNICAS SUPERVISADAS

Técnicas supervisadas

3.1. Introducción y objetivos

Las técnicas supervisadas existen porque los datos pueden considerarse clasificados de acuerdo con alguna variable de interés. Así es posible definir un procedimiento para entrenar un modelo de clasificación, y dicho modelo puede utilizarse como un predictor.

Se puede encontrar en la literatura y específicamente en herramientas estadísticas y de programación librerías que contienen variedad de algoritmos. Existen enfoques para trabajar tanto con datos discretos como continuos. La ventaja de utilizar una herramienta como R project, es aprovechar la capacidad de cómputo y extender su aplicación a datos de gran tamaño.

La generación de modelos de clasificación supervisada a partir de algoritmos evita los procedimientos manuales, las tareas engorrosas de cálculo y la comprensión matemática de técnicas como la regresión, dejando claro que la comprensión matemática del procedimiento es importante en un contexto más avanzado del análisis inteligente de datos. Los modelos pueden ser evaluados, comparados y contrastados en varias iteraciones con diferentes algoritmos mediante métricas de calidad.

Al finalizar este capítulo los lectores estarán en la capacidad de:

- Diferenciar entre técnicas supervisadas y no supervisadas
- Entrenar modelos de clasificación supervisadas
- Evaluar los modelos de clasificación
- Aplicar el modelo de clasificación para fines de predicción y estimación

3.1.1. ¿Qué son las técnicas supervisadas?

De acuerdo con algunos autores en el estudio comparativo de técnicas de aprendizaje supervisado para la clasificación de imágenes utilizando un conjunto de datos, se dice que dicha técnica usa etiquetas en los datos, es decir que son previamente ordenados por los humanos, para posteriormente ser clasificados a

través de una categoría o la regresión para obtener una proyección (Hernández Millán et al., 2018), por ejemplo, para la clasificación de fallos en turbinas de gas para aviones de tipo Jet (Batayev, 2018), estos aprenden de ejemplos etiquetados para poder predecir nuevos datos. El algoritmo de aprendizaje compara la salida con el resultado correcto, encontrando errores para modificar el modelo resultante, es decir que el algoritmo se entrena con un historial de datos y así aprende a asignar la etiqueta de salida adecuada a un nuevo valor (Silva & Bernardino, 2022).

3.1.2. ¿Por qué existen las técnicas supervisadas?

En la vida real podemos encontrar una gran variedad de datos etiquetados y clasificados, pero pueden representar un desafío si no se tienen mecanismos adecuados para analizarlos, para eso se utilizan las técnicas supervisadas. Dichas técnicas son útiles, por ejemplo, para el reconocimiento de imágenes, ya que estas técnicas sirven para evitar costosas colecciones de datos etiquetados y dominios con pocos estándares de modelos pre entrenados.

Las áreas en las que se pueden emplear las técnicas supervisadas son diversas, pero una en la que comúnmente se aplica es el área de la seguridad. Por ejemplo, las técnicas de aprendizaje supervisado se emplean ampliamente en la detección de fraudes con tarjetas de crédito, debido a que hacen uso de la suposición de que se pueden aprender patrones fraudulentos a partir del análisis de transacciones pasadas (Carcillo et al., 2021).

Las técnicas supervisadas tienen otros ámbitos en los que se pueden aplicar, como el reconocimiento de rostros, que va de la mano con el reconocimiento de imágenes (Happy et al., 2019). Las enfermedades también pueden presentar datos que necesiten ser analizadas, por lo que las técnicas supervisadas ayudan en este ámbito también (Rezayi et al., 2021).

3.1.3. ¿Cuándo se utilizan las técnicas supervisadas?

Una de las áreas ampliamente beneficiadas de las TS es la salud, prediciendo riesgos de enfermedades emergentes, Así a manera de ejemplo una gran cantidad de conjunto de datos puede dividirse de la siguiente manera según (Di Noia et al., 2020):

- Conjunto de entrenamiento, que contiene el 50% del número total de patrones disponibles
- Conjunto de validación, que contiene el 25% del número total de patrones disponibles.
- Equipo de prueba, que contiene el 25% del número total de patrones disponibles.

Los métodos de aprendizaje supervisado se aplican generalmente para el reconocimiento y la clasificación de patrones. Recientemente, hay muchos algoritmos de aprendizaje supervisado que se han utilizado en diferentes campos de la ingeniería (Chakraborty et al., 2020).

3.2. Un enfoque de clasificación de las TS

Las técnicas de minería de datos pueden dividirse según el objetivo que persigue, así se tiene predictivos / descriptivos.

Un modelo predictivo responde preguntas sobre datos futuros, mientras que un modelo descriptivo proporciona información sobre las relaciones entre los datos y sus características.

Según (Pérez & Santín, 2008) las predictivas y descriptivas se emplean para el descubrimiento, mientras que las técnicas auxiliares se emplean para la verificación. Además, indican que todas estas técnicas pueden ser utilizadas mediante algoritmos probados e implementados en soluciones de minería de datos. La Figura 15 ilustra la clasificación según los autores citados.

Figura 15

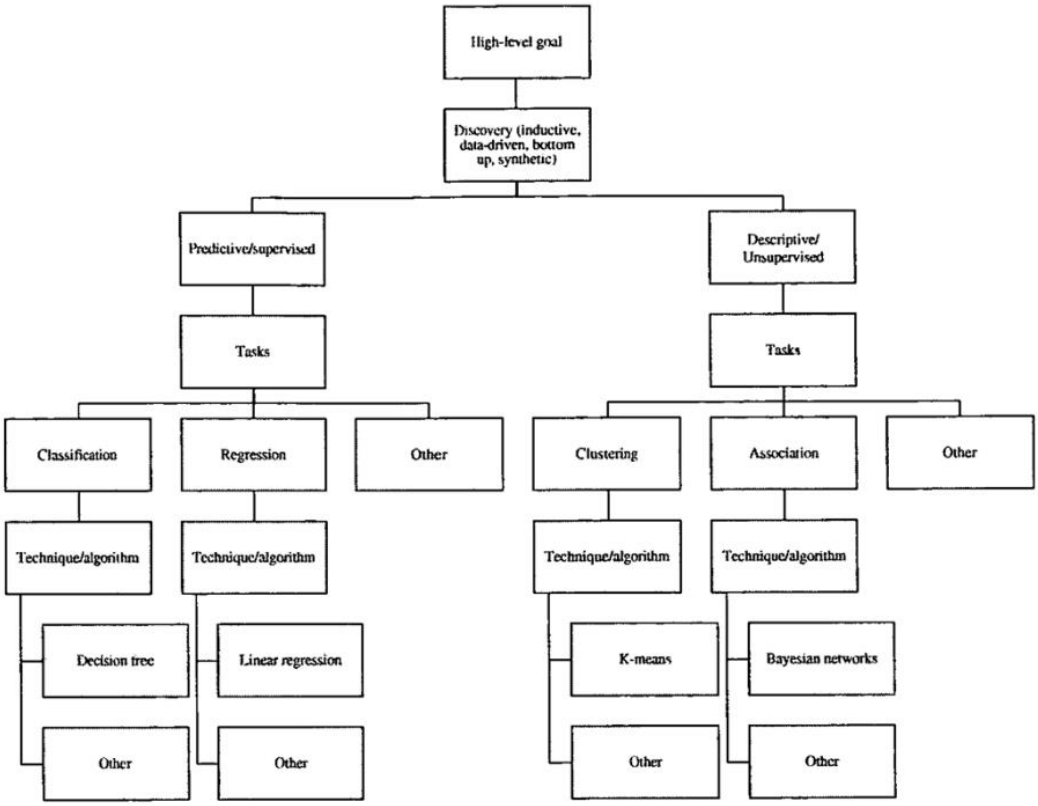
Clasificación de técnicas de minería de datos

Nota: Autores (2023)

En la Figura 16 se pretende reflejar con mayor precisión lo que muchos expertos consideran la extracción de datos "verdadera". Cabe señalar que incluso algunos expertos parecen dar a entender que la extracción de datos "verdadera" tiene una naturaleza predictiva y, por lo tanto, también podría excluirse la extracción de datos descriptivos de la figura. Sin embargo, no es fácil separar las tareas de minería de datos descriptiva y predictiva porque, la minería de datos descriptiva puede servir como base para la minería de datos predictiva (Colonna, 2013).

Figura 16

Una taxonomía desde el enfoque de minería de datos



Nota: Autores (2023)

3.3. Redes neuronales

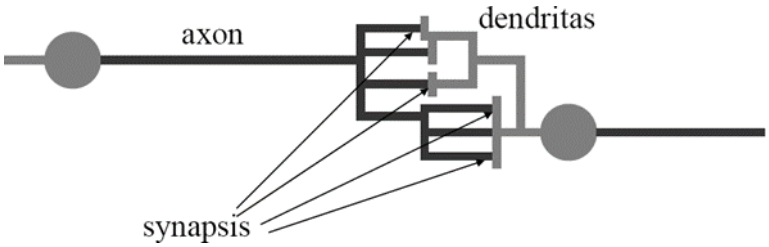
3.3.1. Definiciones

Desde el punto de vista biológico, una neurona es una célula formada por un área engrosada que contiene el núcleo, una prolongación larga llamada axón, y unas prolongaciones más cortas llamadas dendritas.

La transmisión de la información se hace en la sinapsis. El procesamiento de las señales nerviosas incluye dos tipos de fenómenos: eléctricos y químicos. El proceso eléctrico propaga una señal en el interior de la neurona, y el proceso químico trasmite la señal desde una neurona a otra.

Figura 17

Representación de una neurona con sus partes elementales

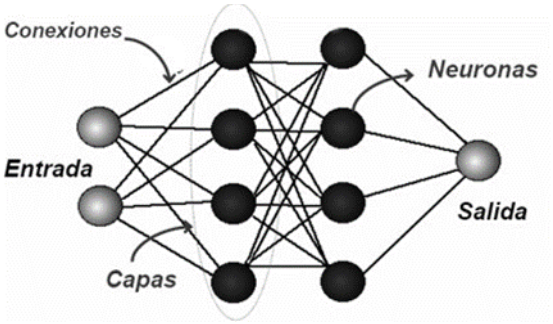


Nota: Autor (2023)

Las redes neuronales artificiales (RNA) (ejemplo en la Figura 18) son estructuras de procesamiento paralelo, que están inspiradas en el funcionamiento neurobiológico humano, de esta forma pretenden emular sus capacidades de aprendizaje, clasificación, reconocimiento, entre otras.

Figura 18

Red neuronal artificial



Nota: Autor (2023)

Su unidad básica de procesamiento es el nodo o neurona: Esta consiste fundamentalmente en una función matemática arbitraria, cuya función es recopilar la información que recibe, procesar esta información y comunicarla a otros nodos.

Para diseñar una red neuronal es necesario definir su estructura externa y los parámetros internos que la caracterizan, en los aspectos externos consta:

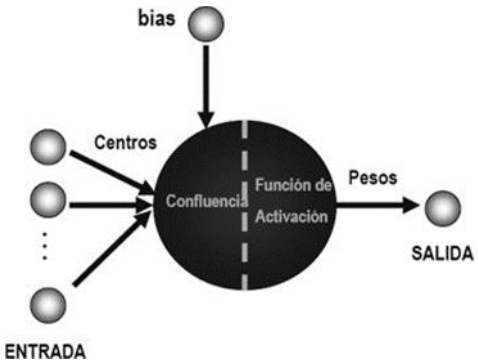
- Número de capas
- Número de neuronas por capa
- Tipo de conexión entre neuronas

Mientras que en los aspectos internos (Figura 19) están:

- Función de Activación: Es la función generadora de la salida de cada neurona, la cual se aplica a la confluencia
- Función de Confluencia: Entrega una señal globalizada de todas las entradas, para esto pondera cada entrada a través de un factor denominado centro de la red, el que valoriza la importancia de la conexión.

Figura 19

Elementos internos de una RNA



Nota: Autor (2023)

A continuación, se describe un ejemplo de la aplicación de Redes neuronales usando R Project.

3.3.2.Caso Práctico: estimación y predicción con redes neuronales (NNET)

En este caso se va a hacer uso de la librería MASS, que contiene los datos que vamos a utilizar en este caso práctico. La librería está instalada por defecto en R.

Los datos que utilizaremos se denominan Boston. Estos datos se corresponden con el censo de 1970 y contienen precios y otras variables relativas al área metropolitana de Boston. En primer lugar, cargamos el conjunto de datos y revisamos sus características.

```
# Carga la librería MASS
library(MASS)
data(Boston)
```

```
boston=Boston
summary(Boston)
```

Primero se aplica un modelo lineal. Este modelo veremos que es demasiado simple (es decir, no es suficientemente complejo/adecuado para este problema). Utilizaremos para ello la función lm (lm significa linear model). Vamos a intentar explicar la variable número 14 (medv: mediana en miles de dólares de los precios de pisos ocupados por propietarios) en función de las otras 13 variables:

```
# Aplicamos la función lm
boston.lm = lm(formula = medv ~., data = boston)
summary(boston.lm)
```

```
Call:
lm(formula = medv ~ ., data = boston)

Residuals:
     Min       1Q   Median       3Q      Max
-15.595   -2.730   -0.518    1.777   26.199

Coefficients:
              Estimate   Std.Error  T value  Pr(>|t|)
(Intercept)  3.646e+01  5.103e+00    7.144  3.28e-12  ***
crim        -1.080e-01  3.286e-02   -3.287  0.001087  **
zn           4.642e-02  1.373e-02    3.382  0.000778  ***
indus        2.056e-02  6.150e-02    0.334  0.738288
chas         2.687e+00  8.616e-01    3.118  0.001925  **
nox         -1.777e+01  3.820e+00   -4.651  4.25e-06  ***
rm           3.810e+00  4.179e-01    9.116  < 2e-16   ***
age          6.922e-04  1.321e-02    0.052  0.958229
dis         -1.476e+00  1.995e-01   -7.398  6.01e-13  ***
rad          3.060e-01  6.635e-02    4.613  5.07e-06  ***
tax         -1.233e-02  3.760e-03   -3.280  0.001112  **
ptratio     -9.527e-01  1.308e-01   -7.283  1.31e-12  ***
black        9.312e-03  2.686e-03    3.467  0.000573  ***
lstat       -5.248e-01  5.072e-02  -10.347  < 2e-16   ***
```

Signif. Codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-squared: 0.7406, Adjusted R-squared: 0.7338
F-statistic: 108.1 on 13 and 492 DF, p-value: < 2.2e-16

Un R^2 de 73.3%, da a entender que el modelo lineal explica bastante la variable respuesta. Las variables “indus” y “age” parecen no tener relación lineal con la variable respuesta. Esto se ve porque no tienen asterisco al lado del p-valor, lo cual es un modo gráfico de indicar que el p-valor es alto, es decir, no podemos descartar la hipótesis nula de que su peso en el modelo de regresión es nulo.

El último p-valor (p-value: < 2.2e-16) es un p-valor conjunto que indica que el grupo de variables sí tiene influencia en la variable respuesta; en otras palabras, alguna de las variables del conjunto de regresores es influyente en la variable respuesta.

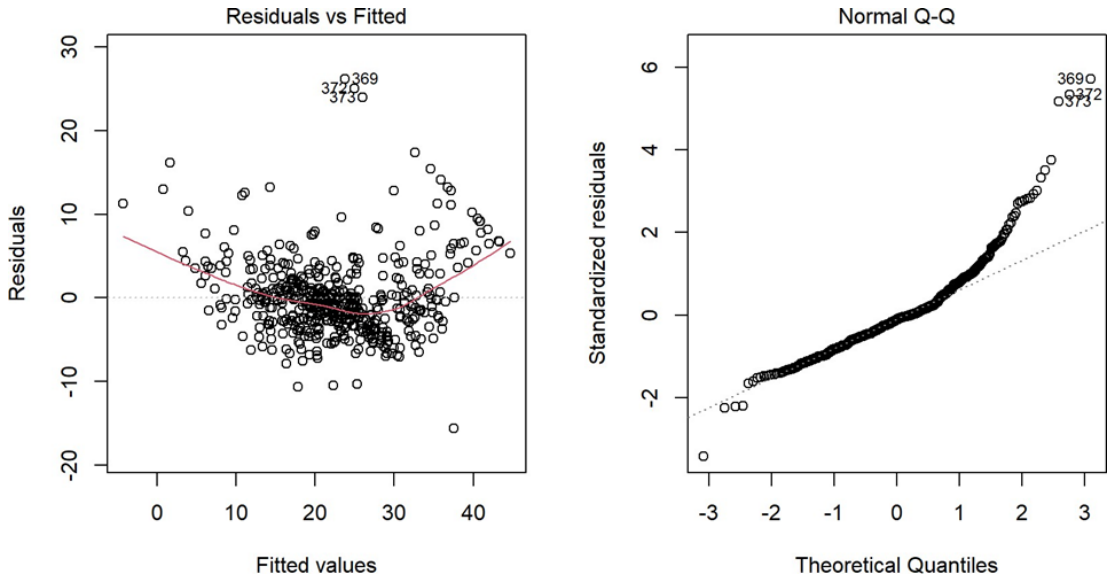
Sin embargo, este modelo lineal sólo es válido si se verifican las hipótesis del modelo de regresión lineal, es decir, normalidad en los residuos y linealidad. Utilicemos las rutinas de R para verificar estos supuestos:

El comando plot(boston.lm) nos muestra secuencialmente todos los gráficos de diagnosis del modelo. Podemos presentarlos de uno en uno así:

```
# Validamos el modelo lineal
plot(boston.lm,1)
plot(boston.lm,2)
```

Figura 20

Gráficos de validación para el modelo de regresión lineal



Nota: Autor (2023)

Es evidente la relación no lineal en el gráfico (Residuos vs valores ajustados), lo cual invalida de entrada el modelo de regresión para este problema: la relación entre el precio de las casas y las variables explicativas es no lineal. También se verifica en la gráfica Normal Q-Q que los residuos no siguen la distribución normal. Por tanto, otra hipótesis fundamental violada, así el modelo de regresión lineal no es en absoluto fiable en este problema.

Ahora utilizaremos una red neuronal para este propósito. En R disponemos de varias librerías entre las que podemos destacar nnet, neural y neuralnet. Utilizaremos la primera:

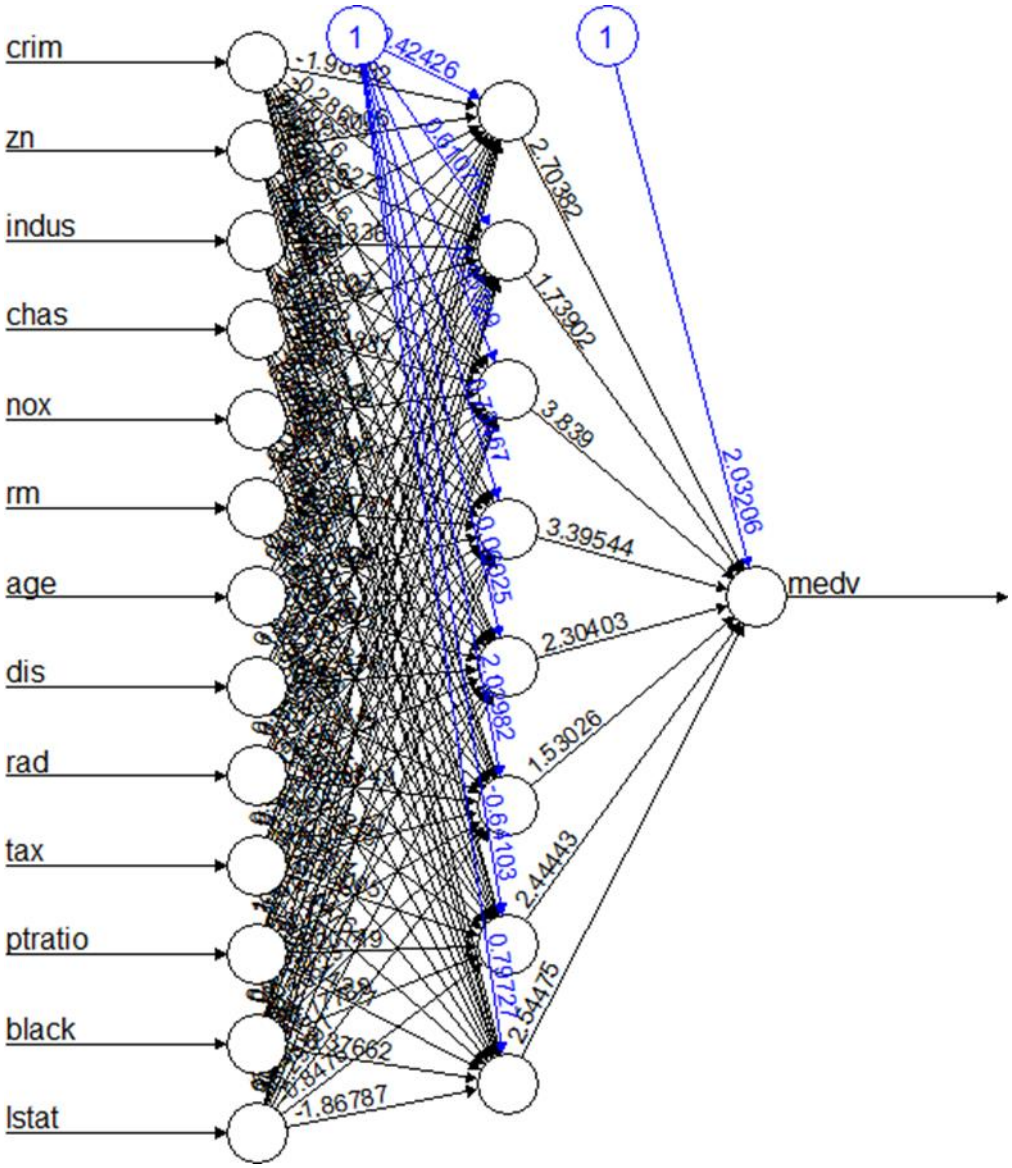
Si el paquete/librería existe, se utiliza el comando library(nnet). Caso de obtener un error por parte del sistema, eso significa que la librería no está instalada en el sistema. Se puede instalar la librería nnet o cualquier otra desde la interface Package dando click en Install.

```
# ejecución de la función de red neuronal
library(nnet)
boston.nnet = nnet(formula = medv ~., data = boston,rang=0.1,size=8,linou
t=T,maxit=1000)boston=Boston
```

```
# weights:121
initial                 value 297211.300754
iter                10  value 34451.446105
iter                20  value 33270.014260
Iter                30  value 32882.529141
Iter                40  value 32549.811119
Iter                50  value 32377.084417
Iter                60  value 32327.387935
.
.
.
Iter               330  value 7843.484412
Iter               340  value 7842.973897
Iter               350  value 7842.367356
Iter               360  value 7841.998319
Iter               370  value 7841.705277
Iter               380  value 7841.400859
Iter               390  value 7841.177197
final                   value 7841.173238
converged
```

Para ver la topología de la red, a continuación, un ejemplo de ejecución concreta. Para ello cargamos la librería neuralnet. De nuevo, si el paquete/librería existe, se utiliza el comando library (neuralnet).

```
# para mostrar la gráfica de la topología de la red
library(neuralnet)
red.boston = neuralnet(medv~.,boston,hidden=8,rep=1)
plot(red.boston)
```



En la rutina nnet, con 8 neuronas ocultas se obtuvo un error de 7841.17. Debido a la naturaleza del algoritmo, que tiene un paso de inicialización de pesos aleatorio, y debido a la existencia de múltiples mínimos locales en la superficie de error, suele ocurrir que en dos ejecuciones para los mismos parámetros obtengamos resultados distintos. Por ello, es conveniente ejecutar varias veces el algoritmo y quedarse con la salida que de menor error.

Por ejemplo, vemos que no siempre sale lo mismo:

```
# cuatro ejecuciones suscesivas, resultados de convergencia diferentes
library(nnet)
boston.nnet = nnet(formula = medv ~., data = boston,rang=0.1,size=8,linou
t=T,maxit=1000)boston=Boston
```

#Ejecución 1

weights: 121

initial value 296280.766421

final value 42716.295415 converged

#Ejecución 2

weights: 121

initial value 299301.537927

iter 10 value 42285.299506

final value 39159.695071 converged

#Ejecución 3

weights: 121

initial value 298931.290486

iter 10 value 35135.323132

final value 34918.528290 converged

#Ejecución 4

weights: 121

initial value 304505.976908

iter 10 value 34005.989098

iter 20 value 31268.314811

iter 30 value 29596.038845

.

.

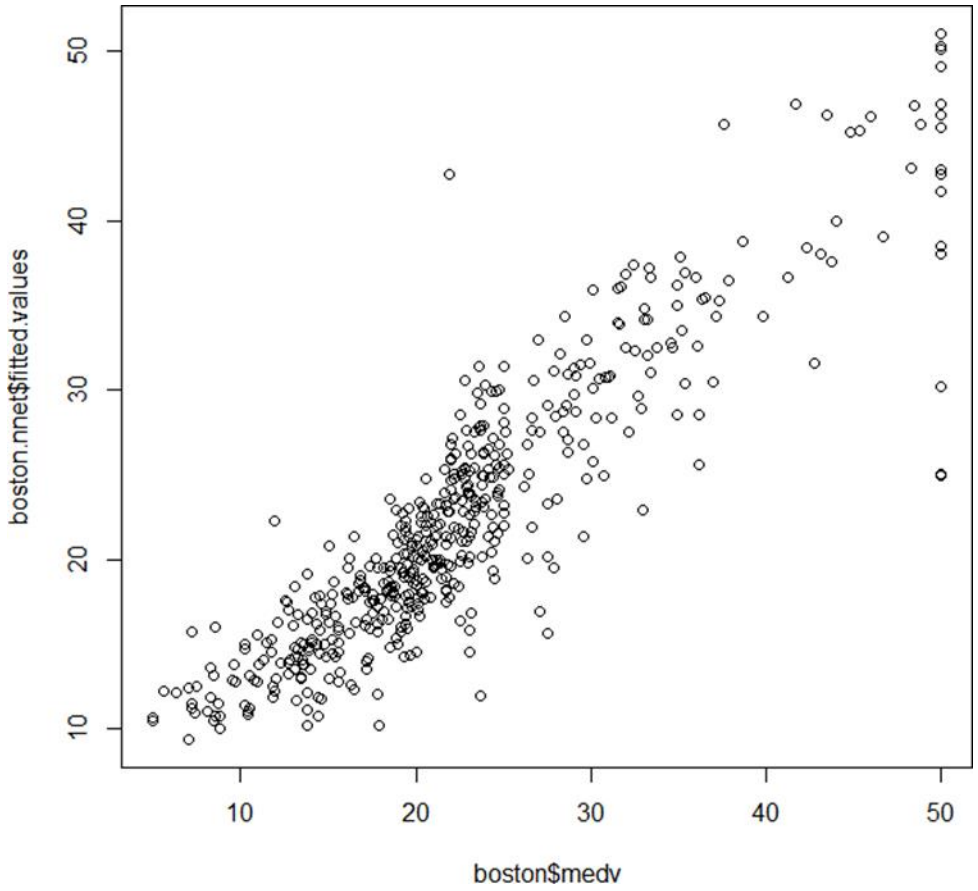
.

iter 450 value 7841.476249

final value 7841.471605

Nos quedamos con la de menor error, y vamos a ver si la red ha ajustado bien los valores de los precios.

```
# Gráfico de valores originales versus valores ajustados
plot(boston$medv,boston.nnet$fitted.values)
```



Como vemos, sale una relación lineal (que es x = y) indicando que el ajuste es bueno. Sin embargo, vemos que la variabilidad no es constante, lo que indica que la predicción es mejor para precios bajos que para precios altos, lo cual es razonable, pues los precios más altos dependen a veces de factores no medibles como prestigio, capricho, fama de los antiguos propietarios, etc.

¿Cómo mejorar? Incrementando el número de unidades en la capa oculta, que es el modo de controlar la complejidad del modelo en neural net.

Pasamos a utilizar 9 unidades y tocamos un parámetro de optimización, decay.

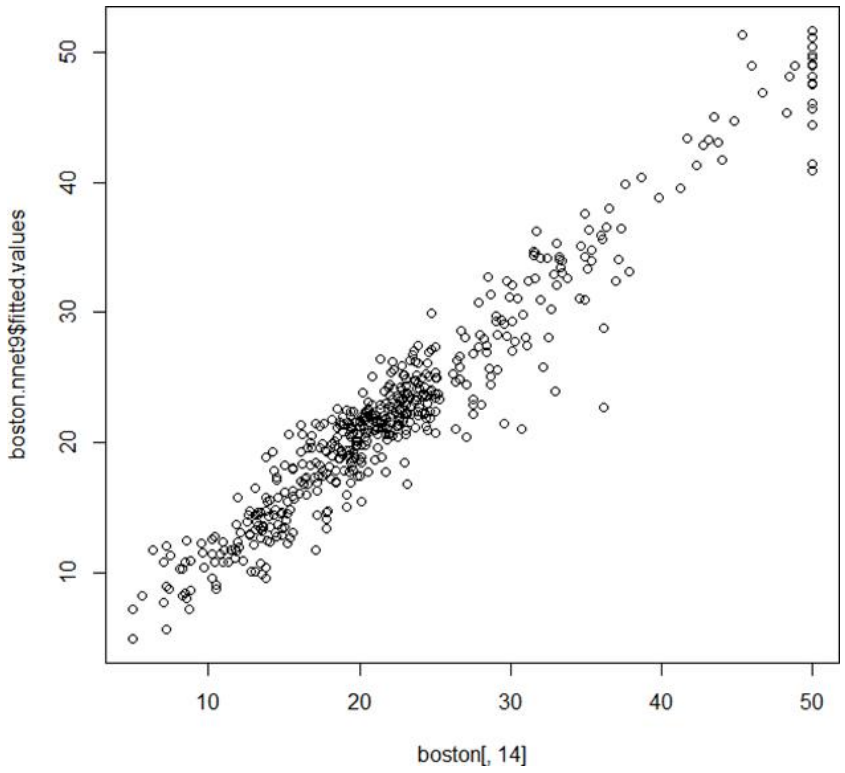
```
# Obtenenos un nuevo modelo ajustando parámetros
boston.nnet9 = nnet(formula = medv ~., data = boston,rang=0.5,size=9,lino
ut=T,decay=10e-2,maxit=2000)
```

```
 # weights: 136
 initial                 value 295395.023266
 iter          10        value 42784.799481
 iter          20        value 33670.044050
```

```
iter        30     value  31695.757918
.                  value
.                  value
.                  value
iter      1030     value  1820.019785
iter      1040     value  1820.007544
iter      1050     value  1819.990315
iter      1060     value  1819.976641
final              value  1819.976419
converged
```

Nuevamente comprobamos la calidad del ajuste, comparamos los valores predichos por la red con los valores reales de la variable respuesta:

```
# Gráfico de valores originales versus valores ajustados
plot(boston$medv,boston.nnet9$fitted.values)
```



Es aparente que la predicción ha mejorado mucho en términos de precisión. Si miramos la correlación entre los valores reales y los predichos, tendremos una medida análoga al R^2 en regresión:

```
> cor(boston[,14],boston.nnet$fitted.values) #primer ajuste [,1]

[1,] 0.9035646

> cor(boston[,14],boston.nnet9$fitted.values) #segundo ajuste [,1]

[1,] 0.9817125
```

Como vemos, la variabilidad explicada es del 98%: la red neuronal con 9 neuronas ocultas es un excelente predictor del precio de las casas para las variables explicativas dadas.

3.3.3. Construcción de un modelo predictivo con NNET utilizando KFOLD-CV

Para el siguiente ejemplo utilizaremos el conjunto de datos basketball, este dataset lo podemos encontrar disponible en "KEEL-dataset repository" (https://sci2s.ugr.es/keel/datasets.php).

Este conjunto de datos contiene una serie de estadísticas (5 atributos) sobre 96 jugadores de baloncesto, los atributos se definen a continuación.

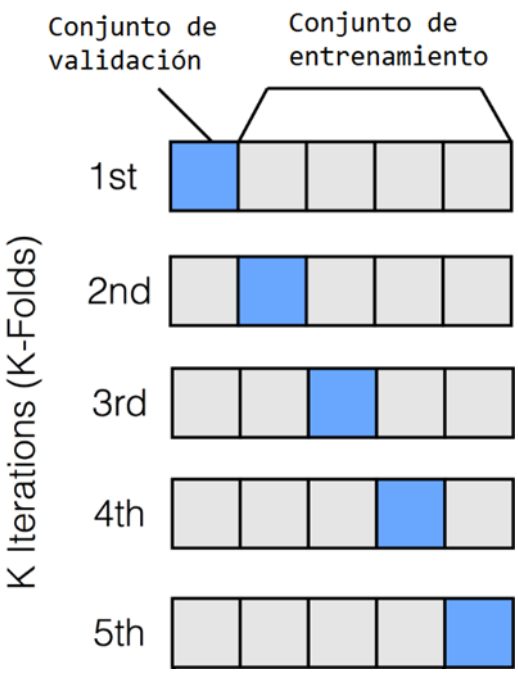
1) assists_per_minuteReal: promedio de asistencias por minuto.
2) heightInteger: altura del jugador.
3) time_playedReal: tiempo jugado por el jugador.
4) ageInteger: número de años del jugador.
5) points_per_minuteReal: promedio de puntos por minuto.

Nos planteamos el siguiente caso, se quiere encontrar un mejor modelo que sirve para predecir la variable "points_per_minuteReal**",** para ello se ejecutan dos fases. Primero, preparación del conjunto de entrenamiento y prueba (método kFold-CV) y segundo, selección de manera elitista del mejor modelo desde los bloques entrenados.

La validación cruzada de k iteraciones o k-fold cross validation (kFold-CV) consiste en dividir los datos originales en k subconjuntos y, al momento de realizar el entrenamiento, se va a tomar cada k subconjunto como conjunto de prueba del modelo, mientras que el resto de subconjuntos (k-1) se tomará como conjunto de entrenamiento (Ver Figura 21).

Figura 21

Representación lógica de método K-fold cross validation



Nota: Autores (2023)

```
# neural net una técnica para trabajar en estimación con variables
# continuas
# ejemplo con el conjunto de datos "basketball"
library(neuralnet)
library(cvTools)
# Carga del dataset basketball
fname= "…//basketaball.csv"
datBkt <- read.csv(fname, header=T)
# Aqui se ha utilizado una función denominada cvFolds
# para construir los bloques de entrenamiento y prueba con 10 folds
# El dataset se divide en 10 bloques 90% para entrenamiento
# 10% para prueba
k<-10
folds <- cvFolds(NROW(datBkt), K=k)
```

En la segunda etapa de la resolución de este problema se procede a evaluar cada uno de los bloques (folds) combinación de entrenamiento-prueba para

alcanzar el modelo de mejor calidad. Este proceso lo hemos incluido dentro de un ciclo de k iteraciones.

```
# mediante un for empieza el recorrido de los bloques de entrenamiento y prueba.
mejor <- 0 for(i in i:k){conjEntrena <- datBkt[folds$subsets[folds$which!=i], ]
# se obtiene conjunto entrenamiento de bloques complemento de i
conjValida <- datBkt[folds$subsets[folds$which==i],]
# se obtiene conjunto validación con el bloque i
# obtener modelo con el conjunto de entrenamiento
new.nnet <-neuralnet(formula=points_per_minuteReal~.,data=conjEntrena,hidden=5,rep=2)
new.pred <- predict(new.nnet , conjValida[,-5])
calidadActual <- cor(conjValida$points_per_minuteReal,new.pred)
if(calidadActual > abs(mejor)){
        mejor.nnet <- new.nnet
        mejor.pred <- new.pred
        mejor.entrena <- conjEntrena
        mejor.validacion <- conjValida
        mejor.i <- i
        mejor <-c
        }
}
resultado=cbind(best.validacion$points_per_minuteReal,best.pred,abs(best.validacion$points_per_minuteReal - best.pred))
```

Dentro del ciclo “for” se prepara el modelo para cada combinación de bloques para entrenamiento y prueba, la siguiente parte del código se encarga de la evaluación del modelo y calcular la calidad del ajuste. En caso de que el ajuste de la combinación de bloques (entrenamiento-prueba) actual sea mayor que las anteriores, entonces actualiza el mejor ajuste junto con otras variables útiles.

	Validación	Predicho	Error
1	0.5885	0.5387703	0.04972972
68	0.3007	0.3550683	0.05436828
70	0.2894	0.4589801	0.16958008
33	0.4903	0.4779111	0.01238885
93	0.2381	0.3012637	0.06316374
51	0.3418	0.4211992	0.07939922
29	0.4325	0.4607427	0.02824269
21	0.4280	0.4848117	0.05681171
4	0.5772	0.5651974	0.01200257

La visualización de “resultado” reúne a las variables Validación (datos originales de prueba), Predicho (datos obtenidos por el modelo seleccionado) y Error (varianza entre el valor original y el predicho). Un cálculo adicional nos da un MSE de 0.016 y una correlación de 90%, por lo cual asumimos que hemos conseguido un buen modelo.

```
> MSE
[1] 0.01626767

> cor(resultado$Validacion,resultado$Predicho)
[1] 0.8929023
```

3.4. Máquinas de soporte vectorial (SVM)

3.4.1. Definiciones

En 1992, Vapnik y colaboradores (Boser et al., 1992) propusieron un algoritmo supervisado para la clasificación que desde entonces ha evolucionado hasta convertirse en lo que ahora se conoce como Support Vector Machines (SVM) (Cristianini & Shawe-Taylor, 2000): una clase de algoritmos para clasificación, regresión y otras aplicaciones que representan el estado actual del arte en el campo.

Su origen está asociado con la teoría del aprendizaje estadístico, esta técnica ha demostrado eficiencia en problemas de reconocimiento de caracteres, existen casos de específicos donde ha obtenido tasas de error del 1.1%. Actualmente, las SVM se ven en el marco de los “kernel methods”.

Básicamente, su procedimiento se basada en que, dado un conjunto de datos, el cual lo vamos a representar algebraicamente de la siguiente forma:

$$\{(X_i, Y_i)\}_{i=1}^{n}, y_i \in Y$$

El mecanismo es transformar x_i a un espacio de dimensión mediante una función $\emptyset$. Donde los x_i representan al espacio de entrada y los $f(x)_i$ la salida. Esta transformación es atractiva porque el problema puede verse más sencillo. Sin embargo, calcular en el espacio transformado, es computacionalmente costoso por alta dimensionalidad y la no linealidad.

El “kernel trick” es la solución al inconveniente de alta dimensionalidad, entonces, los datos aparecen sólo como productos escalares, y si somos capaces de calcularlos, no necesitamos conocer explícitamente la transformación ϕ. Eso ocurre para la mayoría de los algoritmos estadísticos.

Definimos la función kernel K por:

$$k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

En la práctica de las SVM, el usuario especifica la función kernel, y no ϕ(.). La función kernel puede ser interpretada como una medida de similitud entre los datos.

Kernel lineal	$\boldsymbol{k(x, z) = x^T z}$
Kernels polinomiales con grado d	$k(x, z) = (c + x^T z)^d$
Funciones de base radial con base y	$k(x, z) = e^{-y\|\|x-z\|\|^2}$

Tanto los kernels polinomiales como los de base radial llevan los datos a un “espacio de variables” de alta dimensión.

3.4.2.Caso práctico utilizando SVM

Con la finalidad de desarrollar un modelo, se obtienen mediante cálculo las densidades de fabricación exactas en los 314 puntos de una viga. Así, de las 314 densidades calculadas para la fabricación de cada viga, 110 datos coincidirán con los cálculos entregados y los otros 204 se corresponderán con las densidades restantes (las que no se han podido calcular). Los datos proporcionados por el centro externo están en un archivo de datos. La columna 1 contiene la pulgada para la que se realiza el cálculo, y la columna 2 contiene los cálculos para la viga.

Para implementar el procedimiento, se opta por el uso del paquete estadístico R. En este caso se va a hacer uso de la librería e1071, que contiene las rutinas relativas a SVM. Si el paquete existe, se utiliza el comando library(e1071). Caso contrario debe proceder con la instalación del paquete.

```
#Caso práctico basado en algoritmo SVM
library(e1071) # Carga la librería e1071
nombre = ".../datasets/datosSVM.csv"
datosSVM = read.table(nombre)
#creamos variables con nombres familiares
pulgadas = datosSVM [,1]
viga = datosSVM [,2]
summary(datosSVM[,1:2])
plot(pulgadas,viga,type="l")
```

En la vista de resultado se tiene una vista de las principales medidas descriptivas de las dos variables que se ha seleccionado para el ejercicio.

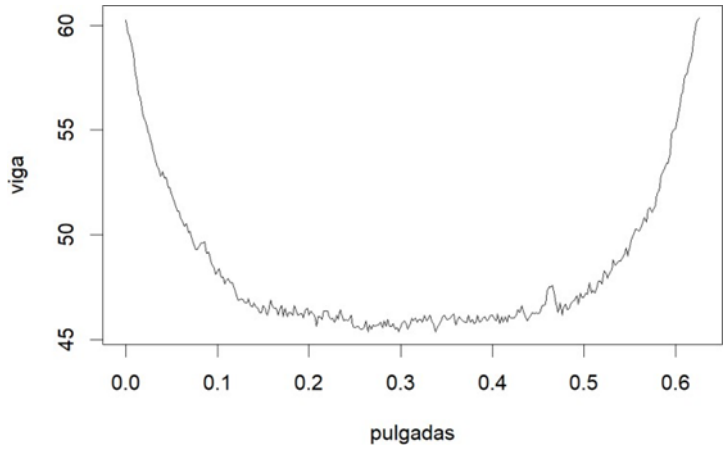
```
> summary(datosSVM[,1:2])
      pulgadas            viga1
 Min.   :0.0000   Min.   :45.36
 1st Qu.:0.1565   1st Qu.:46.02
 Median :0.3130   Median :46.55
  Mean   :0.3130   Mean   :48.37
 3rd Qu.:0.4695   3rd Qu.:49.56
```

Max. :0.6260 Max. :6032

En la Figura 22 se muestra una gráfica de cómo se distribuye la densidad con que se debe fabricar cada viga.

Figura 22

Distribución de puntos de densidad en la viga con todos los datos



Nota: Autores (2023)

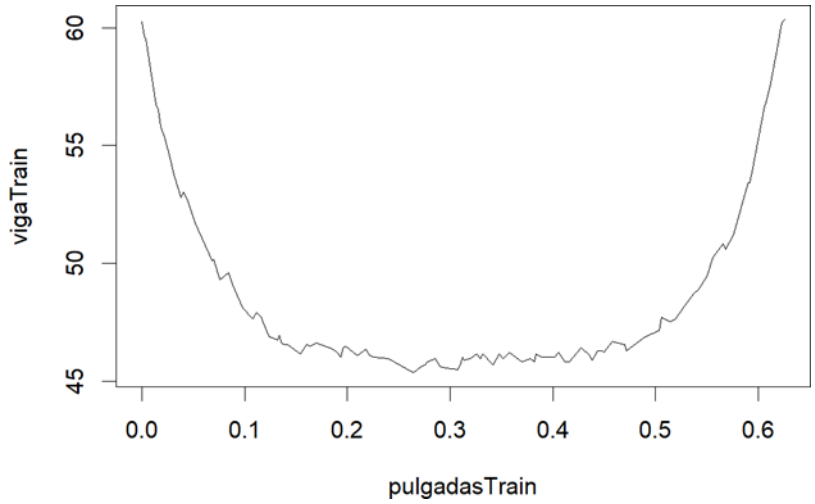
Considerando las condiciones de trabajo real habrá 110 puntos disponibles calculados por los expertos, y se debe estimar los restantes. Para ello se elige al azar 110 puntos y se entrena una SVM con esos datos. Una vez ajustada la SVM, se estima el valor de la densidad para los 204 puntos restantes y se comprueba la calidad de la estimación.

```
# obtenemos el conjunto de entrenamiento y prueba
idxTrain = sample(1:314,110)
idxTrain = sort(idxTrain)
datosSVM.Train=datosSVM[idxTrain,c(1,2)]
datosSVM.Test=datosSVM[-idxTrain,c(1,2)]
pulgadasTrain=datosSVM.Train[,1]
vigaTrain=datosSVM.Train[,2]
# obtenemos una nueva vista gráfica de la muestra
plot(pulgadasTrain,vigaTrain,type="l")
```

Se observa un comportamiento similar en la gráfica generada a partir de la muestra de datos seleccionada (ver Figura 23). Esto no servirá para entrenar un modelo basado en SVM.

Figura 23

Distribución de puntos de densidad en la viga con la muestra



Nota: Autores (2023)

Se procede con el modelo de SVM no lineal, con parámetros Epsilon = 2, C = 100, y un kernel radial con gamma = 40.

```
# obtenemos el modelo con los parámetros especificados
vigas.svm=svm(pulgadasTrain,vigaTrain,type="eps-regression",
              kernel="radial",epsilon=0.02,gamma=40,cost=100,scale=F)
```

3.5. Naïve Bayes

3.5.1. Definiciones

Este clasificador se construye en base al teorema de Bayes (Russell & Norvig, 1995), algunos estudios que comparan algoritmos de clasificación han encontrado que el rendimiento de un clasificador bayesiano simple puede ser comparable con el de un árbol de decisión y clasificadores de redes neurales. Los clasificadores bayesianos también han exhibido alta precisión y velocidad cuando se aplican a bases de datos grandes (Jiawei et al., 2012).

El algoritmo da una forma de combinar la probabilidad previa y las probabilidades condicionales en una sola fórmula, que se pueden usar para calcular la probabilidad de cada una de las posibles clasificaciones a su vez. Hecho esto se elige la clasificación con el valor más grande (Sokolova & Lapalme, 2009).

La probabilidad de que un evento x ocurra en un conjunto de datos es calculada usando la frecuencia de la ocurrencia del evento x en el conjunto de datos dividido por el número total de instancias se conoce como probabilidad previa. En cambio, la probabilidad de que ocurra un evento si sabemos que un atributo tiene un valor particular (o que varios atributos tienen valores particulares) se denomina probabilidad condicional de que ocurra el evento.

Básicamente el algoritmo NB se basa en el siguiente proceso: Dado un conjunto de k clasificaciones mutuamente excluyentes y exhaustivas c_1, c_2, $\cdots$ c_k, que tienen probabilidades previas $P(c_1)$, $P(c_2)$, $\cdots$, $P(c_k)$ respectivamente, y n atributos a_1, a_2, $\cdots$ a_n, que para una instancia dada tienen valores v_1, v_2, $\cdots$ v_n, respectivamente. Puede demostrarse que la probabilidad posterior de la clase c_i que ocurre para la instancia especificada es proporcional a:

Ecuación 7

$$P(c_i)xP(a_1 = v_1 y\ a_2 = v_2 \ldots y\ a_n = v_n|c_i)$$

Lo que es igual a la Ecuación 8, que es una representación matemática más técnica y simplificada de la Ecuación 7.

Ecuación 8

$$P(c_i)x\prod_{j=1}^{n} P(a_j = v_j|clase = c_i)$$

Cuando se utiliza el algoritmo de NB para clasificar una serie de nuevas instancias, la manera más eficaz de comenzar es calcular todas las probabilidades previas y también todas las probabilidades condicionales que implican a un atributo, aunque no todas pueden ser necesarias para clasificar una instancia particular.

Hay que tener presente que al usar este tipo de algoritmo es obviamente necesario que todos sus atributos sean del tipo categórico y que el estimar las

probabilidades por frecuencias relativas puede dar una mala estimación si el número de instancias con una combinación de atributo/valor dada es pequeña.

3.5.2. Construcción de un clasificador Bayesiano

1) Asumir que se quiere predecir la variable Y que asume ny valores distintos y que estos valores son: $v1, v2, \cdots, vny$
2) Asumir que hay m atributos de entrada llamados $x1, x2, \cdots, xm$
3) Dividir el conjunto de datos en ny subconjuntos de datos llamados $DS1, DS2, \cdots, DSny$
4) Definir DSi = Registros en los cuales $Y = vi$
5) Para cada grupo DSi, usamos estimación de densidad para estimar el modelo Mi que modela la distribución de las variables de entrada o entre los registros $y = vi$.
6) Mi estima la función de probabilidad conjunta por clase $P(X1, X2, \cdots, Xm|y = vi)$
7) Para predecir la clase a la cual pertenece el nuevo vector de entradas $((X1 = u1, X2 = u2, \cdots, Xm = um)$ es mejor hallar la clase $y = vi$ para la cual la probabilidad $P(y = vi|X1, X2, \cdots, Xm)$ sea la mayor posible

El clasificador Naïve Bayes puede ser aplicado también cuando hay predictoras continuas, aquí hay dos alternativas

1) Aplicar previamente un método de discretización tal como: Usando intervalos de igual ancho, usando intervalos con igual frecuencia, ChiMerge,1R y la discretización usando el método de la entropía. Todos ellos están disponible en la librería dprep (ver disc.mentr, disc.ew, disc.ef y chiMerge).
2) Asumiendo una distribución para cada predictora, por lo general Gausiana, con media y varianza estimada de los datos. La librería e1071 de R contiene una función naiveBayes que calcula el clasificador Naïve Bayes tanto para datos discretos como continuos.

3.5.3. Ejemplo clasificador Naïve Bayes en datos discretos

Dado el siguiente conjunto de datos (Figura 24)

Figura 24

Ejemplo de conjunto de datos discretos

X1	X2	X3	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	1
1	1	1	1
0	0	1	1
1	1	0	1

Nota: Autores (2023)

$$P(Y=0)=\frac{3}{7} \qquad P(Y=1)=\frac{4}{7}$$

¿A qué clase será asignada el registro $(\mathrm{X}_1=0,\mathrm{X}_2=0,\mathrm{X}_3=1)$?

$$P(X_1=0,X_2=0,X_3=1|Y=0)=P(X_1=0|Y=0)P(X_2=0|Y=0)$$

$$P(X_3=1|Y=0)=\left(\frac{2}{3}\right)\left(\frac{1}{3}\right)\left(\frac{1}{3}\right)=\frac{2}{27}$$

$$P(X_1=0,X_2=0,X_3=1|Y=1)=P(X_1=0|Y=1)P(X_2=0|Y=1)$$

$$P(X_3=1|Y=1)=\left(\frac{2}{4}\right)\left(\frac{2}{4}\right)\left(\frac{3}{4}\right)=\frac{3}{16}$$

Como $\left(\frac{3}{7}\right)\left(\frac{2}{27}\right)<\left(\frac{4}{7}\right)\left(\frac{3}{16}\right)$ entonces $(X_1=0,X_2=0,X_3=1)$ será asignado a la clase 1. Si el objeto está asignado a la clase 0 entonces el NB comete un error.

3.5.4. Ejemplo clasificador Naïve Bayes en datos continuos

Figura 25

Ejemplo de conjunto de datos con variable continua

X1	X2	X3	X4	Y
0	0	1	3.15	0
0	1	0	8.17	0
1	1	0	5.72	0
0	0	1	7.16	1
1	1	1	9.32	1
0	0	1	12.81	1
1	1	0	15.48	1

Nota: Autores (2023)

Mediante discretización consiste en aplicar uno de los métodos de discretización a las variables continuas y entonces se puede resolver por procedimiento discreto. A continuación, se muestra una de las funciones empleadas para discretizar usando R Project sobre el conjunto de datos (Figura 25).

```
# Método mediante discretización de variable continua
dsEjemplo.disc=disc.ef(dsEjemplo,4:5,2)
dsEjemplo.disc
```

El resultado del procedimiento muestra en la variable cuatro dos valores discretos.

```
> dsEjemplo.disc
  X1 X2 X3 X4 Y
1  0  0  1  1 0
2  0  1  0  1 0
3  1  1  0  1 0
4  0  0  1  1 1
5  1  1  1  2 1
6  0  0  1  2 1
7  1  1  0  2 1
```

La librería e1071 de R Project contiene la función que implementa el método NB, además de otras funciones para clasificación. Para este ejercicio note que el conjunto de datos puede ser discreto o continuo. En este caso se ha usado el conjunto de datos que contiene la variable continua (Figura 25).

```
# Aplicación de la librería e1071
nbModelo=naiveBayes(Y~.,data=dsEjemplo)
predicho=predict(nbModelo,dsEjemplo[,-5])
table(predicho,dsEjemplo[,5])
```

El resultado se muestra en el cuadro de salida, note que una vez construido el modelo se ha usado la función “predict” para obtener los valores de la variable “Y”. Una matriz muestra la precisión con Error igual a cero.

```
> predicho
[1] 0 0 0 1 1 1 1
Levels: 0 1
> table(predicho,dsEjemplo[,5])

 predicho  0 1
         0 3 0
         1 0 4
```

3.5.5.Caso práctico utilizando Naïve Bayes

Dado el conjunto de datos “seeds_dataset” (Rwzhang, 2018), que corresponden a las mediciones de las propiedades geométricas de los granos pertenecientes a tres variedades diferentes de trigo. Para la recogida de datos se ha utilizado una técnica de rayos X blandos y un paquete GRAINS para construir los siete atributos de valor real. Se quiere obtener un modelo predictivo para lo cual se ha de probar el clasificador Naïve Bayes con el conjunto de datos proporcionado, se sugiere utilizar una tabla de confusión para evaluar la calidad.

Básicamente la experimentación consiste en tres etapas para obtener el modelo de clasificación mediante el clasificador NB.

1) Extracción, Transformación Y Carga De Datos
2) Construcción Del Modelo

3) Evaluación Del Modelo

En la etapa 1

- Una carga limpia de los datos al entorno de R Project, con el uso de las funciones disponibles y la posterior verificación que hayan sido cargados adecuadamente.

En la etapa 2

- Los datos serán segmentados así: 70% de los datos para el entrenamiento y 30% para la prueba.
- Una vez obtenidos los conjuntos procederemos a construir el modelo mediante el uso de la función disponible en la librería e1071 de R.

En la etapa 3

- Una vez obtenido el modelo procedemos a validar con el conjunto de prueba, como estos modelos entregarían valores discretos de la variable dependiente no hace falta el uso de una función de transformación a valor discreto (esto debido a que existen técnicas que estiman valores reales).
- Con los valores ya discretos se procedería a evaluar mediante el uso de la técnica de matriz de confusión y a calcular las medidas solicitadas.

El conjunto de datos tiene 210 observaciones y 7 atributos continuos, además una variable de clase con tres tipos de trigo (“kama”, “rosa” y “Canadian”). Para construir los datos, se ha medido siete parámetros geométricos de los granos de trigo (área A, perímetro P, compacidad C = 4*pi*A/P^2, longitud del núcleo, ancho del grano, coeficiente de asimetría y longitud del surco del grano). La tabla 5 muestra un resumen del conjunto de datos.

Tabla 5

Resumen conjunto de datos “sedes”

variedad	Atributo	Min	Max	Media	desviación
kama	Area	11.23	17.08	14.33	1.21
	perimetro	12.63	15.46	14.29	0.57
	compasidad	0.83	0.91	0.88	0.016
	long_nucleo	4.9	6.05	5.50	0.23

	ancho_grano	2.85	3.68	3.24	0.17
	coef_asim	0.76	6.68	2.66	1.17
	long_surco	4.51	5.87	5.08	0.26
rosa	Area	15.38	21.18	18.33	1.43
	perimetro	14.66	17.25	16.21	0.62
	compasidad	0.84	0.91	0.88	0.015
	long_nucleo	5.36	6.67	6.14	0.26
	ancho_grano	3.23	4.03	3.67	0.18
	coef_asim	1.47	6.68	3.64	1.18
	long_surco	5.14	6.55	6.02	0.25
canadian	Area	15.38	21.18	18.33	1.43
	perimetro	14.66	17.25	16.14	0.62
	compasidad	0.84	0.91	0.88	0.015
	long_nucleo	5.36	6.67	6.15	0.26
	ancho_grano	3.23	4.03	3.67	0.18
	coef_asim	1.47	6.68	3.64	1.18
	long_surco	5.14	6.55	6.02	0.25

Nota: Autores (2023)

En esta experimentación se ha dividido el conjunto de datos de la siguiente manera:

- 70% para entrenamiento
- 30% para prueba

De forma alternativa se puede hacer uso del método “kFold-CV”, en muchos casos es muy recomendable porque se suele obtener modelos de mayor calidad.

```
#Caso práctico basado en algoritmo Naïve Bayes
library(e1071) # Carga la librería e1071
library(caret)
nombre = ".../datasets/seeds.csv" # localización del archivo de datos
seeds_dataset <-read.table(nombre)
# creamos variables con nombres familiares
# Obtenemos conjuntos de entrenamiento y prueba
set.seed(1649)
conjEntrena <- sample_frac(seeds_dataset, .7)
```

```
conjValida <- setdiff(seeds_dataset, conjEntrena)
# construcción del modelo
modelo.nb=naiveBayes(variedad~.,data=conjEntrena)
# aplica el modelo para predecir nuevos valores en variable clase
valores.predNB <- predict(modelo.nb , seeds_dataset[,-8])
# preparamos valores predichos y originales para comparar
valores.predNB =as.factor(c(valores.predNB))
valores.originales=as.factor(c(seeds_dataset[,8]))
metricas.nb = confusionMatrix(data= valores.predNB, reference =
valores.originales)
```

El resultado muestra una precisión del 90% con este método.

```
> metricas.nb

Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 60  5  4
         2  3 65  0
         3  7  0 66

Overall Statistics

               Accuracy : 0.9095

                 95% CI : (0.8623, 0.9446)

    No Information Rate : 0.3333

    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8643

Mcnemar's Test P-Value : NA
```

3.6. Arboles de decisión

3.6.1.3.6.1.Definiciones

Un diagrama o árbol de decisiones es un modelo de la evaluación de una función discreta, en el que se determina el valor de una variable y la siguiente acción (elegir otra variable para evaluar o generar el valor de la función) se elige en consecuencia. Los árboles de decisión y los diagramas (también conocidos como procedimientos de evaluación secuencial) tienen amplias aplicaciones en bases de datos, programación de tablas de cálculo, teoría de la complejidad concreta, teoría de conmutación, reconocimiento de patrones y taxonomía; en resumen, donde quiera que las funciones discretas deban evaluarse secuencialmente (Moret, 1982).

Los árboles de decisión se pueden definir como conjunto de reglas representadas en forma de una estructura de árbol. Son muy útiles cuando hay más de una manera para convertirse en miembro de una clase meta.

A manera de ejemplo un modelo para encontrar tarjetas habientes rentables puede identificar tres tipos de clientes:

- Tarjetas habientes que mantienen saldos altos
- Tarjetas habientes que compran mucho
- Tarjetas habientes que ocasionalmente hacen compras grandes y pagan sus balances a tiempo

Cada uno de estos representa un paso diferente a través del árbol.

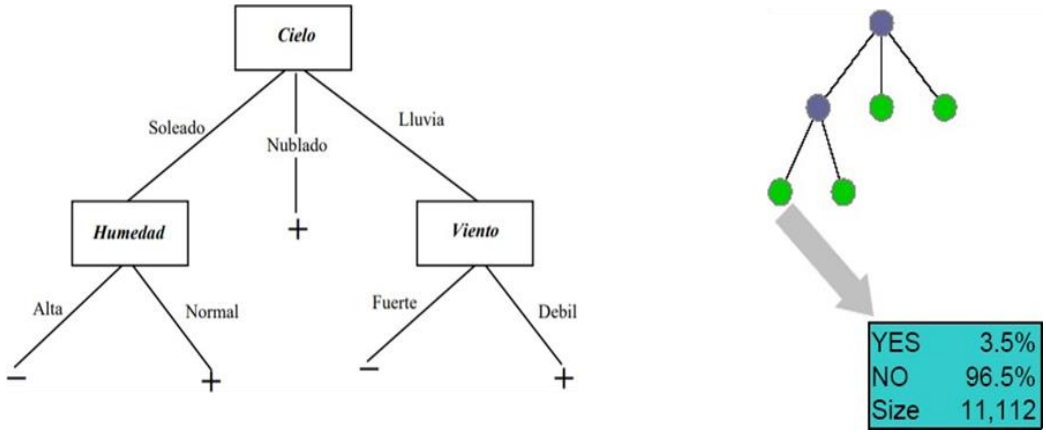
Cada hoja contiene información sobre el número de observaciones que caen en ella y la proporción para cada clase. La clase más densa se selecciona como la clasificación para el nodo.

Se utilizan para

- Asignar “scores” a los datos (Usualmente hay muy pocas hojas por lo que hay pocos valores de “scoring”)
- Explorar datos
- Hacer clasificaciones y predicciones
- Comprender que variables son más importantes

Figura 26

Ejemplos de árboles de decisión



Nota: Autores (2023)

La aplicación más común es en clasificación y predicción, además de contribuir en la selección de variables importantes antes de usar los modelos de regresión. Para la creación de un árbol de decisión se utiliza un conjunto de datos de entrenamiento (“training”) en una primera etapa, luego se utiliza un conjunto de datos de validación para reducir la complejidad del árbol y generalizarlo (proceso de poda o “pruning”), así eliminar el problema del “sobreajuste”.

Tres conjuntos de datos obtenidos de particionar el conjunto de datos original

- Entrenamiento: 40%
- Validación: 30%
- Prueba: 30%

Proceso recursivo

1) Se inicia con todos los datos del conjunto de entrenamiento en la raíz.
2) Para cada variable “input” se decide la mejor forma para separar los valores de la variable “target”.
    a. Se selecciona la variable “input” y criterio de mejor separación mediante ésta para los valores de la variable “target”.
3) Se divide el nodo en cuestión en dos o más hijos de acuerdo con aquella variable que “mejor divide” la variable “target”.
4) Se repite proceso con los otros nodos hasta que no sea posible más divisiones.

Los algoritmos más comunes son:

- CART (“classification and regression trees”)
- C5.0
- CHAID (“chi square automatic induction”)
- Random Forest

3.6.2. Caso Práctico utilizando arboles de decisión

Dado el conjunto de datos “basketball” (KEEL, s/f.), se requiere obtener un modelo de regresión basado en árbol de decisión para estimar la variable “points_per_minuteReal”.

Este conjunto de datos contiene una serie de estadísticas (5 atributos) para 96 jugadores de baloncesto.

1) assists_per_minuteReal: promedio de asistencias por minuto.
2) heightInteger: altura del jugador.
3) time_playedReal: tiempo jugado por el jugador.
4) ageInteger: número de años del jugador.
5) points_per_minuteReal: promedio de puntos por minuto

Tabla 6

Resumen del conjunto de datos “basketball"

Atributo	Min	Max	Media	desviación
assists_per_minuteReal	0.049	0.3437	0.1613	0.0597
heightInteger	160	203	189.9	6.960
time_playedReal	10.08	40.71	25.94	8.6211
ageInteger	22	37	27.74	3.325
points_per_minuteReal	0.1593	0.8291	0.4203	0.1088

Nota: Autores (2023)

Para resolver este caso primero se realiza una carga limpia de los datos, para ello se descarga el conjunto de datos de la dirección web indicada. Se muestra un resumen descriptivo básico, para tener una apreciación general del comportamiento de los datos (Tabla 6).

Similar a otros ejemplos se utiliza el 70% de proporción para entrenamiento y 30% para prueba. Se utilizan los algoritmos “random forest” y “CART” como técnicas basadas en árboles para regresión y se procede a la construcción de los modelos. Una vez completado el modelo se evalúan en la proporción de prueba y alternativamente sobre todo el conjunto de datos.

```
# Ejercicio de árboles con variables continuas, como modelos de regresión
# Conjunto de datos basketball
library(randomForest)) # Carga la librería
library(caret)
nombre = ".../datasets/basketball.csv" # localización de datos
bkt=read.table(fname,sep=",",dec=".",header=TRUE) # Carga de datos
# Obtenemos conjuntos de entrenamiento y prueba
set.seed(1649)
bkt_entrenamiento <- sample_frac(bkt, .7)
bkt_prueba <- setdiff(bkt, bkt_entrenamiento)
# construcción del modelo
modeloRF <- randomForest(formula = points_per_minuteReal~.,
                         data=bkt_entrenamiento)
modeloCART <- rpart(formula = points_per_minuteReal ~ .,
                    data = bkt_entrenamiento)
# Prediccción en Test
PrediccionRF <- predict(modeloRF, bkt_prueba)
PrediccionCART <- predict(modeloCART, bkt_prueba)
# Crea matriz de resultados
Resultado=cbind(original=bkt_prueba[,5],PrediccionRF,PrediccionCART)
cor(Resultado)
# Calculo de error cuadrado medio (MSE)
MSErf=mean((Resultado[,1]-Resultado[,2])^2)
MSE_cart=mean((Resultado[,1]-Resultado[,3])^2)
MSE_df <- data.frame(algoritmo=c("RF","CART"),MSE=c(MSErf,MSE_cart))
```

Con los dos modelos construidos se han obtenido predicciones sobre el conjunto de prueba, una matriz de resultados ha sido generada a partir de las predicciones y el conjunto original.

La correlación entre las predicciones y el conjunto original es una forma de evaluar la calidad del ajuste de cada técnica.

```
> cor(Resultado)
                 original  PrediccionRF  PrediccionCART
original         1.0000000 0.6692612     0.7051079
PrediccionRF     0.6692612 1.0000000     0.8821512
PrediccionCART   0.7051079 0.8821512     1.0000000
```

```
> MSE_df
  algortimo  MSE
1 RF         0.006149088
2 CART       0.004995708
```

En este primer experimento es notable que el mejor modelo corresponde al algoritmo CART con 70%, y un error de 0.005.

Versión alternativa del caso usando KFOLD-CV

Con el objetivo de perfeccionar la búsqueda de un modelo se aplica el método de kFold-CV para alcanzar mayor calidad. Para ello se ha trabajado sobre el mismo ejemplo anterior, pero agregando la implementación de los bloques de entrenamiento y prueba para un valor de k=10. La librería “cvtools” permite el uso de la función “cvFolds” para la construcción de KFOLDS sobre el conjunto de datos.

Para la búsqueda del mejor modelo se ha implementado un procedimiento mediante un ciclo “for”, este bucle permitirá ir variando los conjuntos de entrenamiento y prueba hasta el valor de (k=10). Durante el recorrido de los bloques se generaría 10 modelos, de los cuales se selecciona el mejor de una forma elitista mediante la medida de correlación (calidad).

Con los modelos obtenidos se procede a realizar las predicciones sobre todo el conjunto de datos.

```
# Ejercicio de árboles con variables continuas, como modelos de
# regresión
library(cvTools)
library(randomForest))
library(caret)
nombre = ".../datasets/basketball.csv" # localización de datos
bkt=read.table(fname,sep=",",dec=".",header=TRUE) # Carga de datos
i=1
k=10
folds = cvFolds(NROW(bkt), K=k)
mejor_corRF=0
mejor_corCART=0
# mediante un for empieza el recorrido de los bloques
# de entrenamiento y prueba.
for(i in i:k){
# se obtiene conjunto entrenamiento de bloques complemento de i
conjEntrena = bkt[folds$subsets[folds$which!=i], ]
# se obtiene conjunto validación con el bloque i
conjValida = bkt[folds$subsets[folds$which==i],]
# obtener modelo con el conjunto de entrenamiento
modeloRF = randomForest(formula = points_per_minuteReal~.,
                        data=conjEntrena)
modeloCART = rpart(formula = points_per_minuteReal ~ .,
                        data = conjEntrena)
PrediccionRF = predict(modeloRF, conjValida[,-5])
PrediccionCART = predict(modeloCART, conjValida[,-5])
cRF = cor(conjValida$points_per_minuteReal,PrediccionRF)
cCART = cor(conjValida$points_per_minuteReal,PrediccionCART)
if(cRF > mejor_corRF){
  mejor_RF=modeloRF
  mejor_corRF=cRF
```

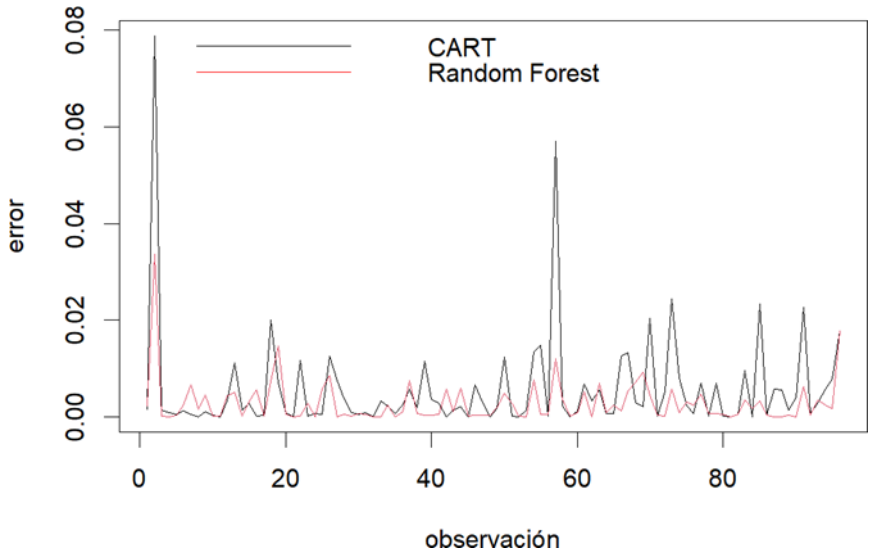
```
  }
if(cCART > mejor_corCART){
   mejor_CART=modeloCART
   mejor_corCART=cCART
  }
}
# Obtiene los valores predichos desde cada modelo creado
PrediccionRF = predict(mejor_RF, bkt[,-5])
PrediccionCART = predict(mejor_CART, bkt[,-5])
Resultado=cbind(original=bkt[,5],PrediccionRF,PrediccionCART)
cor(Resultado)
```

```
> cor(Resultado)
                 original PrediccionRF PrediccionCART
original        1.0000000    0.9166959      0.7160213
PrediccionRF    0.9166959    1.0000000      0.8212376
PrediccionCART  0.7160213    0.8212376      1.0000000

> MSErf=mean((Resultado[,1]-Resultado[,2])^2)
> MSE_cart=mean((Resultado[,1]-Resultado[,3])^2)
> MSE_df <- data.frame(algoritmo=c("RF","CART"),MSE=c(MSErf,MSE_cart))
> MSE_df
  algoritmo       MSE
1        RF 0.002877916
2      CART 0.005771608
```

En el cuadro de resultados se puede observar que en esta versión del problema se tiene como mejor modelo “Random Forest”, que tiene un ajuste del 91% y un error cuadrado medio de 0.0028 (ver Figura 27).

Figura 27

Gráfico de error en ajuste CART y Random Forest



Nota: Autores (2023)

El modelo construido con el uso de kFold-CV ha dado mejores resultados en la construcción del modelo, la recomendación es que se debe prestar mayor detalle a la definición del modelo (evitar particiones simples en los datos, pero también no caer en los sobreajustes).

3.6.3. Caso práctico de comparación en varias técnicas supervisadas

En este caso se tiene la necesidad de pronosticar el resultado final de una contienda electoral, dado que solo el 76.27% de las actas han sido escrutadas y el candidato de la lista A ya ha alcanzado el 38.77% de los votos. El objetivo es predecir el resultado final, es decir, el porcentaje de votos que obtendrá el candidato de la lista A una vez que se hayan escrutado todas las actas.

En base a la problemática presentada, se plantea la necesidad de pronosticar el resultado de una contienda electoral a partir de un conjunto de datos que indica el porcentaje de votos obtenidos por el candidato de la lista A en un determinado porcentaje de actas escrutadas. Para ello, se propone realizar un análisis de clasificación utilizando cinco algoritmos diferentes: regresión lineal, MARS, SVM, Random Forest y Naïve Bayes.

La experimentación se divide en tres etapas, las cuales se describen a continuación:

1) **Extracción, Transformación y Carga de Datos.** En esta etapa, se realiza una carga limpia de los datos al entorno de R Project, asegurándose de que hayan sido cargados adecuadamente. Se realiza una exploración inicial de los datos para identificar posibles errores o inconsistencias.
2) **Construcción de los Modelos.** Los datos se dividen en dos conjuntos: uno de entrenamiento, que corresponde al 70% de los datos, y otro de prueba, que corresponde al 30% restante. Con el conjunto de entrenamiento se procede a construir los modelos utilizando las funciones disponibles en las librerías e1071, earth y randomForest de R para cada uno de los cinco algoritmos considerados.
3) **Evaluación del Modelo.** Una vez obtenidos los modelos mediante las etapas de extracción, transformación y carga de datos y construcción de los modelos, se procede a evaluar su desempeño en el conjunto de datos de prueba. Para evaluar los modelos se utilizará la siguiente métrica:
    - Error cuadrático medio (MSE): mide el error cuadrático promedio entre los valores reales y los valores predichos por el modelo.

Para cada uno de los algoritmos utilizados (regresión lineal, MARS, SVM, Random Forest y Naïve Bayes), se obtendrán los valores predichos para el conjunto de datos de prueba y se calcularán las métricas de evaluación mencionadas anteriormente.

El resumen de votaciones.csv muestra información estadística sobre dos variables: **escrutado** y **votos_listaA**. La variable **escrutado** representa el porcentaje de actas escrutadas, con un mínimo de 10.17% y un máximo de 76.27%. La variable **votos_listaA** representa el porcentaje de votos obtenidos por el candidato de la lista A, con un mínimo de 37.95% y un máximo de 38.77%. La mediana de **votos_listaA** es 38.14%, lo que significa que el 50% de las actas escrutadas tienen un porcentaje de votos para la lista A por debajo de ese valor. En general, los valores de ambas variables están muy cercanos entre sí, lo que sugiere que hay una alta correlación entre ellas.

```
> summary(data_votaciones)
    escrutado         votos_listaA
Min.   :10.17   Min.   :37.95
1st Qu.:26.27   1st Qu.:37.99
Median :42.37   Median :38.14
Mean   :42.94   Mean   :38.21
3rd Qu.:60.17   3rd Qu.:38.38
Max.   :76.27   Max.   :38.77
```

En primer lugar, importamos todas las librerías necesarias y cargamos el conjunto de datos.

```
# Librerias utilizadas
library(e1071)
library(caret)
library(ggplot2)
library(earth)
library(randomForest)
## Conjuto de datos votaciones
fname = choose.files() # Seleccionamos el archivo.
data_votaciones = read.csv(fname, header = T)
# Construimos los datos de entrenamiento y otro de prueba
set.seed(123)
indice_entrenamiento = sample(1:nrow(data_votaciones),
0.7*nrow(data_votaciones))
datos_train = data_votaciones[indice_entrenamiento, ]
datos_test = data_votaciones[-indice_entrenamiento, ]
```

Luego, dividimos los datos en conjuntos de entrenamiento y prueba. A continuación, construimos varios modelos utilizando diferentes algoritmos de aprendizaje automático. Para evaluar el rendimiento de cada modelo, calculamos el error cuadrático medio (MSE) y lo presentamos en una tabla resumen.

El siguiente código generara los modelos y realización de una predicción con el conjunto de datos de prueba.

```
# Construimos los modelos y realizamos una predición con los datos
# de prueba con los diferentes algoritmos.
# Regresión lineal
lm_model <- lm(votos_listaA ~ escrutado, data = datos_train)
pred_lm <- predict(lm_model, newdata = datos_test)
# MARS
mars_model <- earth(votos_listaA ~ escrutado, data = datos_train)
pred_mars <- predict(mars_model, newdata = datos_test)
# SVM
svm_model <- svm(votos_listaA ~ escrutado, data = datos_train)
pred_svm <- predict(svm_model, newdata = datos_test)
# Random Forest
rf_model <- randomForest(votos_listaA ~ escrutado, data = datos_train)
pred_rf <- predict(rf_model, newdata = datos_test)
# Naïve Bayes
nb_model <- naiveBayes(votos_listaA ~ escrutado, data = datos_train)
pred_nb <- predict(nb_model, newdata = datos_test)
```

Es necesario calcular el error cuadrado medio para cada algoritmo y condensarlo en una tabla.

```
# Calculamos el MSE para todos los algoritmos
# MSE Regresión lineal
mse_lm <- mean((pred_lm - datos_test$votos_listaA)^2)
# MSE MARS
mse_mars <- mean((pred_mars - datos_test$votos_listaA)^2)
# MSE SVM
mse_svm <- mean((pred_svm - datos_test$votos_listaA)^2)
# MSE Random Forest
mse_rf <- mean((pred_rf - datos_test$votos_listaA)^2)
# MSE Naïve Bayes
pred_nb_numeric = as.numeric(as.character(pred_nb))
mse_nb <- mean((pred_nb_numeric - datos_test$votos_listaA)^2)
# Construimos una tabla con los valores MSE
tabla_mse <- data.frame(
```

```
  Modelo = c("Regresión lineal", "MARS", "SVM", "Random Forest", "Naïve
Bayes"),
  MSE = c(mse_lm, mse_mars, mse_svm, mse_rf, mse_nb)
  )
```

Con el código anterior obtenemos como resultado una tabla con todos los valores MSE de los modelos.

```
> tabla_mse
```

	Modelo	MSE
1	Regresión lineal	0.006391880
2	MARS	0.000165194
3	SVM	0.003692537
4	Random Forest	0.004117706
5	Naïve Bayes	0.181200000

La tabla muestra los valores del Error Cuadrático Medio (MSE) para cada modelo construido con diferentes algoritmos de aprendizaje automático. El MSE es una medida de la calidad de ajuste de un modelo, siendo un valor más bajo indicativo de una mejor predicción. Se observa que el modelo MARS tiene el menor MSE, lo que sugiere que es el modelo que mejor se ajusta a los datos y predice con mayor precisión. Por otro lado, el modelo de Naïve Bayes tiene un MSE significativamente mayor que los demás modelos, lo que indica que su capacidad de predicción es la menos precisa. Los resultados de la tabla pueden ser utilizados para elegir el mejor modelo para el pronóstico de votos en una contienda electoral. MARS es un algoritmo que ha dado buenos resultados en varias experimentaciones, un trabajo interesante se realizó en (Jaramillo, Villarroel-Molina, et al., 2021), donde resultó un método bastante acertado en un conjunto de datos de gran volumen.

Vamos a realizar la predicción del 100% de los datos con todos los algoritmos, para posteriormente graficarlo y analizar su comportamiento.

```
# Realizamos la predicción de los datos al 100%
# Ajuste final con el modelo de regresión lineal
lm_model_final <- lm(votos_listaA ~ escrutado, data =
data_votaciones)
datos_completos_lm <- data.frame(
    escrutado = 1:100,
    votos_listaA = predict(lm_model_final, newdata =
data.frame(escrutado = 1:100)))
# Ajuster final con el modelo de MARS
mars_model_final <- earth(votos_listaA ~ escrutado, data =
data_votaciones)
datos_completos_mars <- data.frame(
    escrutado = 1:100,
    votos_listaA = predict(mars_model_final, newdata =
data.frame(escrutado = 1:100)))
# Ajuste final con el modelo de SVM
svm_model_final <- svm(votos_listaA ~ escrutado, data =
data_votaciones)
datos_completos_svm <- data.frame(
    escrutado = 1:100,
    votos_listaA = predict(svm_model_final, newdata =
data.frame(escrutado = 1:100)))

# Ajuste final con el modelo de Random Forest
rf_model_final <- randomForest(votos_listaA ~ escrutado, data =
data_votaciones)
datos_completos_rf <- data.frame(
    escrutado = 1:100,
    votos_listaA = predict(rf_model_final, newdata =
data.frame(escrutado = 1:100)))
# Ajuste final con el modelo de Naïve Bayes
nb_model_final <- naiveBayes(votos_listaA ~ escrutado, data =
data_votaciones)
datos_completos_nb <- data.frame(
```

```
    escrutado = 1:100,
    votos_listaA = predict(nb_model_final, newdata =
data.frame(escrutado = 1:100)))
datos_completos_nb$votos_listaA =
as.numeric(as.character(datos_completos_nb$votos_listaA))
```

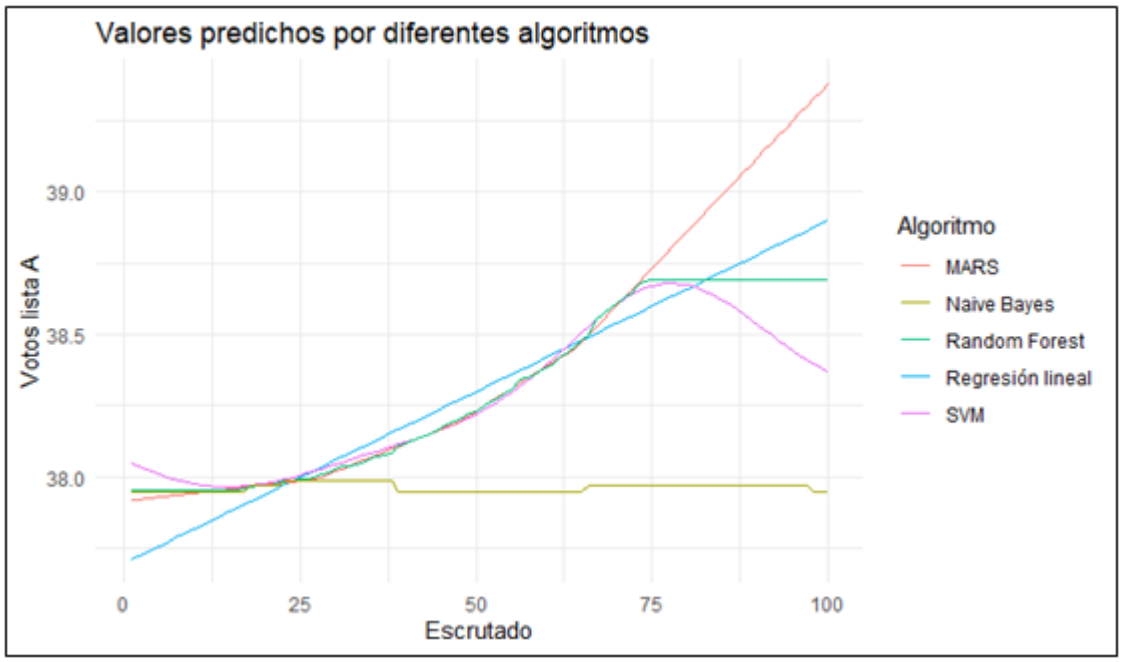
Ahora generamos un gráfico de líneas para representar el comportamiento de los valores predichos.

```
# Generando una gráfica de línea.
ggplot() +
    # Datos completos del modelo de regresión lineal
    geom_line(data = datos_completos_lm, aes(x = escrutado, y =
votos_listaA, color = "Regresión lineal")) +
    # Datos completos del modelo SVM
    geom_line(data = datos_completos_svm, aes(x = escrutado, y
= votos_listaA, color = "SVM")) +
    # Datos completos del modelo MARS
   geom_line(data = datos_completos_mars, aes(x = escrutado, y
= votos_listaA, color = "MARS")) +
    # Datos completos del modelo Random Forest
    geom_line(data = datos_completos_rf, aes(x = escrutado, y =
votos_listaA, color = "Random Forest")) +
    # Datos completos del modelo Naïve Bayes
    geom_line(data = datos_completos_nb, aes(x = escrutado, y =
votos_listaA, color = "Naive Bayes")) +
    # Personalización de la leyenda y etiquetas de los ejes
    labs(title = "Valores predichos por diferentes algoritmos",
        x = "Escrutado",
        y = "Votos lista A",
        color = "Algoritmo") +
theme_minimal()
```

Con el código anterior obtenemos la Figura 28 la cual representa la comparación del rendimiento de los algoritmos de clasificación MARS, Naïve Bayes, Random

Forest, Regresión Lineal y SVM. La línea correspondiente al algoritmo MARS muestra una tendencia clara y constante a lo largo del escrutinio. Sin embargo, las líneas de los otros algoritmos no siguen esta tendencia y permanecen constantes en la mitad de la gráfica. Este resultado sugiere que el algoritmo MARS puede ser más adecuado para este problema en particular en comparación con los otros algoritmos evaluados.

Figura 28
Valores predichos de la lista A



Nota: Autores (2023)

04 CAPITULO

TÉCNICAS NO SUPERVISADAS

Técnicas no supervisadas

4.1. Introducción y objetivos

Las técnicas no supervisadas trabajan sobre datos sin etiquetar y se centran en descubrir patrones y estructuras ocultas sin una guía explicita. En este capítulo se abordará las técnicas no supervisadas como lo son la agrupación, la minería de asociaciones y la reducción de dimensionalidad.

Se proporcionará una guía práctica y accesible para explorar las técnicas no supervisadas y su aplicación en diferentes contextos. Desde la agrupación hasta la reducción de dimensionalidad, las técnicas no supervisadas son una herramienta valiosa en la caja de herramientas de cualquier analista de datos.

Al finalizar este capítulo los lectores estarán en la capacidad de:

- Diferenciar y aplicar las técnicas no supervisadas.
- Describir conjuntos de datos mediante técnicas de agrupación.
- Encontrar relaciones entre los elementos de un grupo de datos.
- Reducir variables basado en su aportación.

4.1.1. ¿Por qué existen las técnicas no supervisadas?

Aunque es cierto que existen grandes cantidades de datos que se encuentran etiquetados hoy en día, también hay que tener en cuenta la continua generación de nuevos datos, los cuales no tienen una etiqueta prestablecida y es difícil o consumirían muchos recursos el asignarles una etiqueta a estos datos. Un ejemplo de estos son las grandes cantidades de datos generados por el tráfico de una red el cual puede analizado mediante técnicas no supervisadas para poder detectar ataques o intrusiones a la red (Eskin et al., 2002).

4.1.2. ¿Cuándo se utilizan las técnicas no supervisadas?

Las técnicas no supervisadas se utilizan cuando se desea descubrir patrones emergentes en los datos, explorar la estructura subyacente y descubrir información valiosa que no es evidente a simple vista. Son particularmente útiles cuando se trabaja con grandes conjuntos de datos, cuando la información

etiquetada disponible es limitada o cuando no se sabe de antemano lo que se está buscando.

4.2. Técnicas de agrupamiento

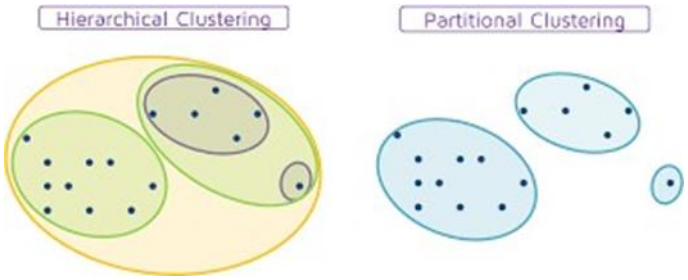
4.2.1. Definiciones

El agrupamiento de datos, también conocido como clustering o clasificación no supervisada, es una técnica de aprendizaje no supervisado utilizada para encontrar patrones en un conjunto de datos sin etiquetas previas. Es útil para la descripción de los datos, pero no para inferir o dar explicación a los datos.

Las aplicaciones de estas técnicas son muy variadas como la investigación de mercados, astronomía, psiquiatría, bioinformática o genética (Everitt et al., 2011). Otro ejemplo de aplicabilidad es la planificación de rutas turísticas mediante una versión mejorada de K-means (Lou, 2022).

Existen varios tipos de técnicas de agrupamiento. Entre las principales tenemos: agrupamiento jerárquico y agrupamiento particionado (ver Figura 29).

Figura 29
Gráfico de proceso de los tipos de agrupamiento



Nota: Autores (2023)

4.2.2. Medidas de similitud

Para encontrar grupos entre las observaciones de los datos y hacer que entre estos grupos sean lo más diferentes posible se hace uso de medidas de similitud. Entre las principales medidas de similitud para las variables cuantitativas encontramos las funciones de cálculo de distancia euclidiana y la distancia de Manhattan (Ramírez et al., 2004). Para las variables categóricas se puede utilizar

las medidas de similitud de Overlap, Eskin, Anderberg (Boriah et al., 2008)(Šulc & Řezanková, 2019).

Distancia Euclidiana: Es la distancia clásica, como la longitud de la recta que une dos puntos en el espacio euclídeo:

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

Distancia de Manhattan: o distancia por cuadras (city-block). Como su nombre indica, hace referencia a recorrer un camino no en diagonal (por el camino más corto), sino zigzagueando, como se haría en Manhattan:

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

Overlap: La medida de superposición cuenta el número de atributos que coinciden en las dos instancias de datos. El rango de similitud por atributo para la medida de superposición es [0,1], con un valor de 0 cuando no hay coincidencia y un valor de 1 cuando los valores de los atributos coinciden.

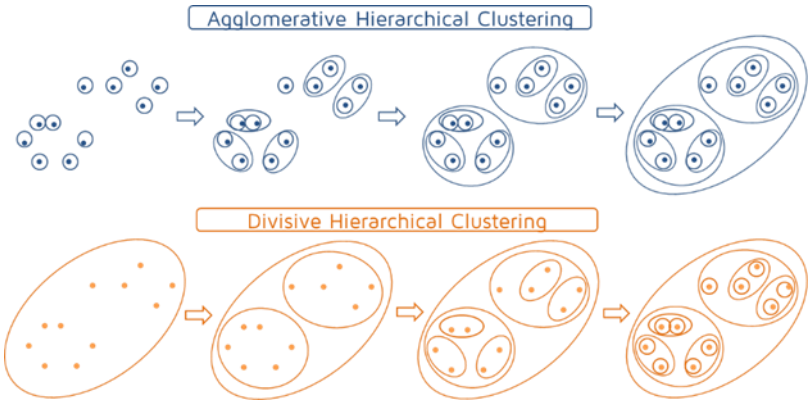
4.2.3. Agrupamiento jerárquico

El agrupamiento jerárquico utiliza algoritmos que buscan agrupar las observaciones de datos en conjuntos o grupos separados, creando una estructura jerárquica de grupos, es decir unos grupos pueden contener a otros subgrupos. Se utilizan cuando de antemano no se conoce la cantidad de grupos o el número de observaciones no es tan grande. Estas técnicas se dividen en dos tipos: aglomerativas y divisivas.

Tal como su nombre lo sugiere, las técnicas de agrupamiento jerárquicos aglomerativas inician tratando a cada observación como un grupo individual y gradualmente une estos grupos individuales en nuevos grupos y así sucesivamente hasta que todas las observaciones formen parte de un solo grupo. En cambio, las técnicas divisivas tratan inicialmente a todas las observaciones como un grupo único y de él van encontrando subgrupos hasta

que los subgrupos finales estén formados por una sola observación. En la Figura 30 se puede ver los grupos y subgrupos en cada tipo de técnica de agrupamiento jerárquicos.

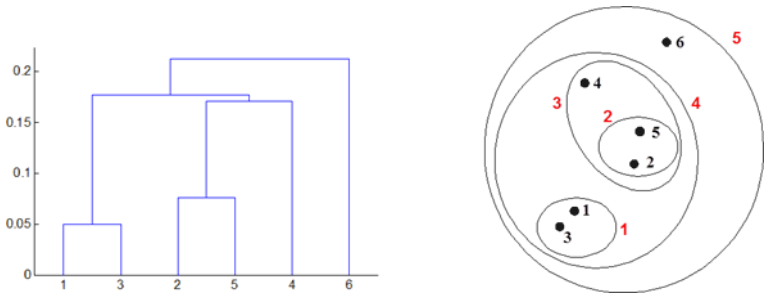
Figura 30
Gráfico del proceso de los tipos de agrupamiento jerárquico



Nota: Autores (2023)

Los resultados de este tipo de análisis es una jerarquía de puntos basada en un árbol que es típicamente representado como un dendrograma (ver Figura 31). En el dendrograma se visualiza los grupos jerárquicos y el analista puede seleccionar el número de grupos de forma subjetiva.

Figura 31
Gráfico de un agrupamiento jerárquico



Nota: Autores (2023)

Caso Práctico: Agrupamiento Jerárquico

En el siguiente caso práctico utilizará el conjunto de datos USArrests (McNeil, 1977) el cual contiene estadísticas, en arrestos por cada 100,000 residentes por agresión, asesinato y violación en cada uno de los 50 estados de EE. UU. en 1973. A este conjunto de datos se realiza un agrupamiento jerárquico

aglomerativo utilizando el paquete de **fastcluster**, este paquete tiene una interfaz para ejecutar el algoritmo propuesto en (Müllner, 2011), el cual resulto ser uno de los más eficientes en comparación con otros de los algoritmos de agrupamiento jerárquico aglomerativo propuestos hasta la fecha, y utiliza como método de aglomeración al enlace promedio (Average-Linkage). El método de enlace promedio (Average- Linkage) define la distancia entre dos grupos como la distancia promedio entre los puntos de datos en el primer grupo y los puntos de datos en el segundo grupo. Sobre la base de esta definición de distancia entre grupos, en cada etapa del proceso combinamos los dos grupos que tienen la distancia de enlace promedio más pequeña.

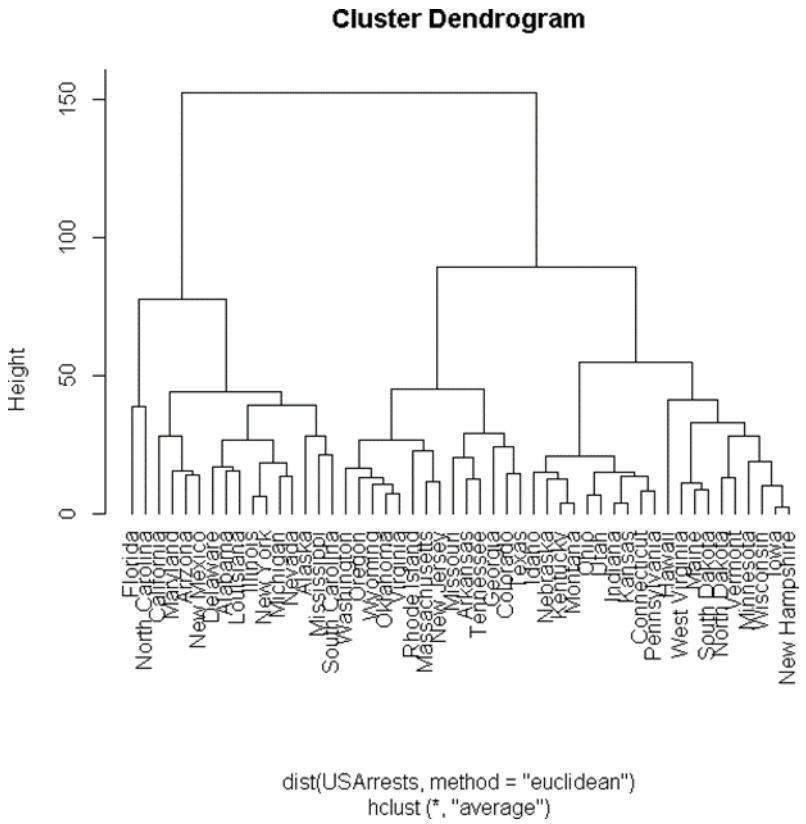
```
#####################################################################
# Agrupamiento Jerárquico Aglomerativo
#####################################################################
# Importa los paquetes necesarios
library(fastcluster)
library(graphics)
# Se carga y se visualiza los datos
data("USArrests")
head(USArrests)
```

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7

```
# Utiliza el conjunto de datos, la distancia euclidiana y el
# método de agregación de grupos Average-Linkage
hc <- hclust(dist(USArrests, method = "euclidean"), "ave")
# Graficamos los cluster representados por el dendrograma
plot(hc, hang = -1)
```

Figura 32

Dendrograma de un agrupamiento jerárquico



Nota: Autores (2023)

De acuerdo con el dendrograma de la Figura 32 se puede ver los agrupamientos jerárquicos realizados en los que pueden ser utilizados para realizar una caracterización de los estados que comparten similitudes de acuerdo con los datos de arrestos y con ello determinar algunas de las medidas que aplicar a estas agrupaciones. Adicionalmente se puede usar el dendrograma obtenido para determinar el número de agrupaciones óptimas para este conjunto de datos es de 2.

4.2.4. Agrupamiento particionado

El agrupamiento particionado utiliza algoritmos que buscan agrupar los datos en conjuntos o grupos separados. Estos algoritmos parten de un número inicial de grupos, y van reorganizando los datos en función de las similitudes o diferencias entre ellos, hasta que se alcanza una configuración de grupos óptima.

Los algoritmos de agrupación examinan los datos para encontrar grupos de elementos que sean similares (Bramer, 2016). Para ello algunas de estas

técnicas utilizan las funciones de distancia antes descritas para determinar los elementos más homogéneos dentro del grupo y lo más heterogéneos entre los grupos.

4.2.4.1. K-MEANS

K-means es un algoritmo propuesto por (Queen, 1967), es rápido y fácil de implementar. Este algoritmo básicamente divide un conjunto de datos de n objetos en k grupos, donde k es un número predefinido de grupos. K-means busca minimizar la suma de las distancias cuadradas entre los puntos y sus centroides en cada grupo. Este algoritmo es ha sido utilizado en diferentes ámbitos y estudios como por ejemplo para analizar el comportamiento de estudiantes (Onofrio et al., 2016), realizar un análisis de sentimientos en un conjunto de datos de revisiones de diferentes productos de Amazon (J. W. Q. L. S. W. Y. Liu, 2019), o para determinar áreas de abstención en las actividades electorales (Wahyuni et al., 2023).

K-means divide los n objetos en k grupos de manera iterativa como se muestra a continuación (M. Chen et al., 2011):

1) Seleccionando k objetos como los núcleos iniciales, cada uno es el núcleo de un grupo;
2) Repita el siguiente proceso (3) y (4) hasta que el resultado deje de cambiar;
3) Calcular la distancia (mediante una medida de similitud) entre cada objeto y los núcleos iniciales, y distribuir cada objeto en un grupo de acuerdo con la distancia. La regla es que la distancia debe ser la más pequeña entre todas las distancias entre el objeto y todos los núcleos iniciales;
4) Recalcular para obtener el objeto central de cada grupo, si su núcleo ha cambiado.

La complejidad computacional de K-means es $O(nkt)$, donde n es el número de objetos, k es el número de grupos y t es la cantidad de iteraciones.

4.2.4.2. Caso Práctico: agrupamiento particionado con K-MEANS

Se utilizará el conjunto de datos USArrests con el cual se realizó anteriormente un agrupamiento jerárquico aglomerativo, pero en esta ocasión se aplicará el algoritmo de agrupamiento particionado K-means. Adicionalmente se empleará el paquete de **factoextra** para calcular el número de k agrupaciones óptimas para el conjunto de datos.

```
#####################################################################
# Agrupamiento Particionado K-means
#####################################################################
# Importa los paquetes necesarios
library("factoextra")
# Se carga y se visualiza los datos
data("USArrests")
head(USArrests)
# Aplica una escala debido a las diferencias de valores
# entre sus variables
datos <- scale(USArrests)
head(USArrests)
```

Muestra el conjunto de datos aplicado la escala en el cual se aplicará el algoritmo K-means.

	Murder	Assault	UrbanPop	Rape
Alabama	1.24256408	0.7828393	-0.5209066	-0.003416473
Alaska	0.50786248	1.1068225	-1.2117642	2.484202941
Arizona	0.07163341	1.4788032	0.9989801	1.042878388
Arkansas	0.23234938	0.2308680	-1.0735927	-0.184916602
California	0.27826823	1.2628144	1.7589234	2.067820292
Colorado	0.02571456	0.3988593	0.8608085	1.864967207

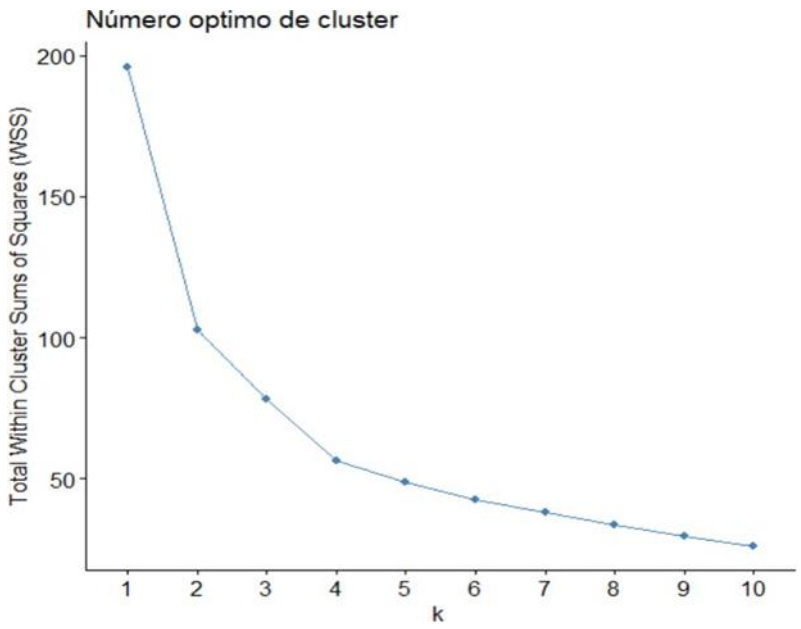
A continuación, se muestra la aplicación del método codo para la determinación de un valor de k optimo, para ello se determina en cual valor de k se tiene una estabilización del valor de **total within sums of squares** (**wss**) (ver Figura 33),

es decir cuando la mejora en el valor de compacidad según wss deja de ser sustancial. En este caso muestra una estabilización en la compacidad en un valor de k = 4.

```
# Utilizar el método de codo para determinar el número
# de k agrupaciones óptimo
fviz_nbclust(datos, FUNcluster = kmeans, method = "wss", k.max = 10,
    diss = get_dist(datos, method = "euclidean"), nstart = 50)+
labs(title= "Número optimo de cluster") +
xlab("k ") +
ylab("Total Within Cluster Sums of Squares (WSS)")
```

Figura 33

Gráfico compactación según wss en función de k agrupaciones



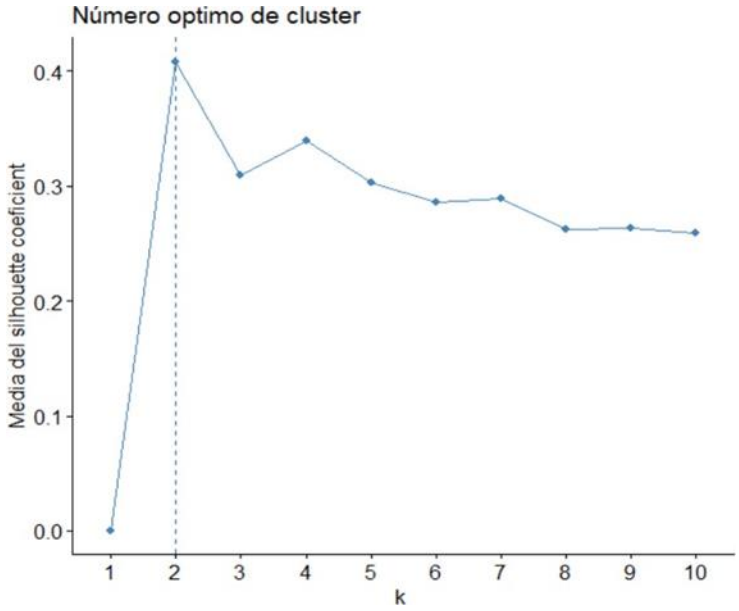
Nota: Autores (2023)

Otro de los métodos para encontrar un valor óptimo de k agrupaciones para un algoritmo de agrupación particionada es el método de **average silhouette** (ver Figura 34). En este método se maximiza la media de los silhouette coeficient (índices silueta), el cual determina cuan buena se ha realizado la asignación de un elemento a una agrupación. En este caso muestra que el valor máximo para el silhouette coeficient se da en un valor de k = 2.

```
# Utilizar el método de average silhouette para determinar
# el número de k agrupaciones óptimo
fviz_nbclust(datos, kmeans, method = "silhouette",k.max = 10,
    diss = get_dist(datos, method = "euclidean"), nstart = 50)+
labs(title= "Número óptimo de cluster") +
xlab("k ") +
ylab("Media del silhouette coeficient")
```

Figura 34

Gráfico la media del índice de silhouette en función de k agrupaciones

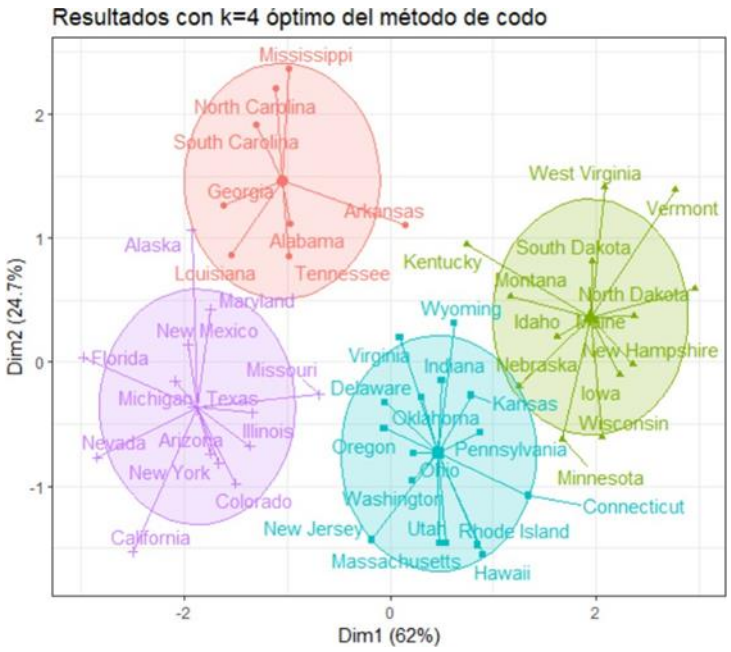


Nota: Autores (2023)

```
# Graficar las k=4 agrupaciones que es el valor óptimo
# según el método de codo
set.seed(123)
km_clusters <- kmeans(datos, centers = 4, nstart = 50)
fviz_cluster(object = km_clusters, data = datos, show.clust.cent = TRUE,
        ellipse.type = "euclid", star.plot = TRUE, repel = TRUE) +
    labs(title = "Resultados con k=4 óptimo del método de codo") +
    theme_bw() + theme(legend.position = "none")
```

Figura 35

Gráfico las k agrupaciones optimas según método de codo

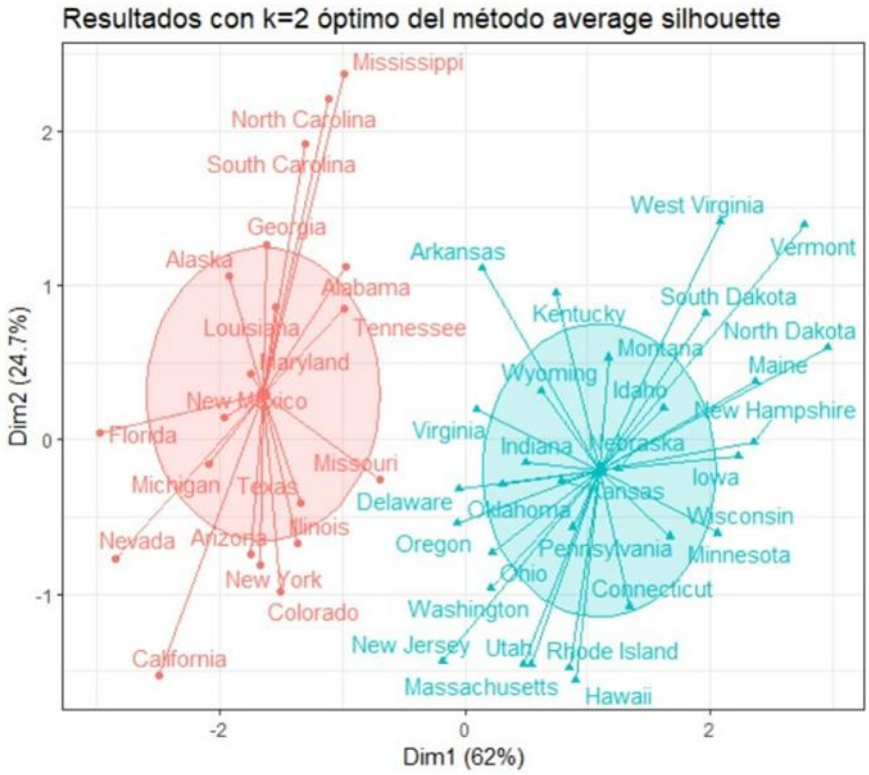


Nota: Autores (2023)

```
# Graficar las k=2 agrupaciones que es el valor óptimo
# según el método de average silhouette
set.seed(123)
km_clusters <- kmeans(datos, centers = 2, nstart = 50)
fviz_cluster(object = km_clusters, data = datos, show.clust.cent = TRUE,
        ellipse.type = "euclid", star.plot = TRUE, repel = TRUE) +
labs(title="Resultados con k=2 óptimo del método average silhouette") +
theme_bw() +
theme(legend.position = "none")
```

Figura 36

Gráfico las k agrupaciones óptimas según método average silhouette



Nota: Autores (2023)

4.3. Reglas de asociación

4.3.1. Definiciones

La minería de patrones frecuentes juega un papel importante en las asociaciones y correlaciones y revela una propiedad intrínseca e importante del conjunto de datos, una forma de expresar estas asociaciones son las reglas de asociación.

Las reglas de asociación son una técnica de minería de datos que se utiliza generalmente para descubrir patrones en conjuntos de datos transaccionales. Estas reglas se basan en la frecuencia con que aparecen diferentes conjuntos de elementos en las transacciones.

Por ejemplo, supongamos que tenemos una base de datos que registra las compras de los clientes en un supermercado, mediante reglas de asociación podemos descubrir patrones en las compras de los clientes, como qué productos

suelen comprar juntos y así organizar las estanterías de acuerdo a estas asociaciones (Agrawal & Srikant, 1994) o en un servidor web podemos conocer cuáles son los itinerarios más seguidos por los visitantes a las páginas web, y entonces, utilizar esta información para estructurar las páginas web en el servidor (Ramírez et al., 2004). Además de los ejemplos anteriores, las reglas de asociación pueden ser útiles en una gran variedad de aplicaciones, como la recomendación de productos (Kutuzova & Melnik, 2018), la segmentación de clientes (Giha et al., 2003) o la detección de diabetes (Amraoui et al., 2019).

Una regla de asociación está compuesta por dos partes: un antecedente y un consecuente. El antecedente es el conjunto de elementos que se utilizan para predecir la ocurrencia del consecuente. Por ejemplo, si el antecedente es "computador" y el consecuente es "software de antivirus", la regla de asociación podría ser "si un cliente compra un computador, entonces también es probable que compre software de antivirus", y se representa de la siguiente forma:

computador ⇒ software_de_antivirus [soporte = 2%, confianza = 60%]

Las reglas de asociación tienen principalmente tres medidas de interés las cuales reflejan la utilidad, la certeza y la asociación del patrón encontrado: cobertura o también llamada soporte (*support*), la confianza (*confidence*) y la elevación (lift). El soporte indica la frecuencia con la que aparece un conjunto de elementos en las transacciones y en el caso de la regla anterior indicaría que el 2% de todas las transacciones muestran que la computadora y el software de antivirus se compran juntos; la confianza indica la probabilidad de que el consecuente aparezca en una transacción dado que el antecedente también aparece y en el caso de la regla plateada nos dice que el 60% de los clientes que compraron una computadora también compraron el software de antivirus (Jiawei et al., 2022); y el lift nos indica la fuerza de la asociación entre dos elementos.

Podemos utilizar diferentes algoritmos, como los algoritmos Apriori, Eclat, FP-growth, para descubrir reglas de asociación en conjuntos de datos. Además, podemos visualizar y analizar las reglas de asociación utilizando gráficos y estadísticas descriptivas. En la literatura también existen otras propuestas de algoritmos con mejoras utilizando algoritmos genéticos o metaheurísticos como MODENAR (Alatas et al., 2008) o QM_VMO (Jaramillo, Garzás, et al., 2021).

4.3.2. Algoritmo Apriori

Algoritmo Apriori (Agrawal et al., 1993) es el primer algoritmo propuesto para encontrar los conjuntos de elementos más frecuentes en una gran base de datos y luego generar asociaciones. Los resultados se presentan en forma de reglas de asociación, que indican la probabilidad de que un conjunto de elementos aparezca junto con otro conjunto de elementos en una transacción.

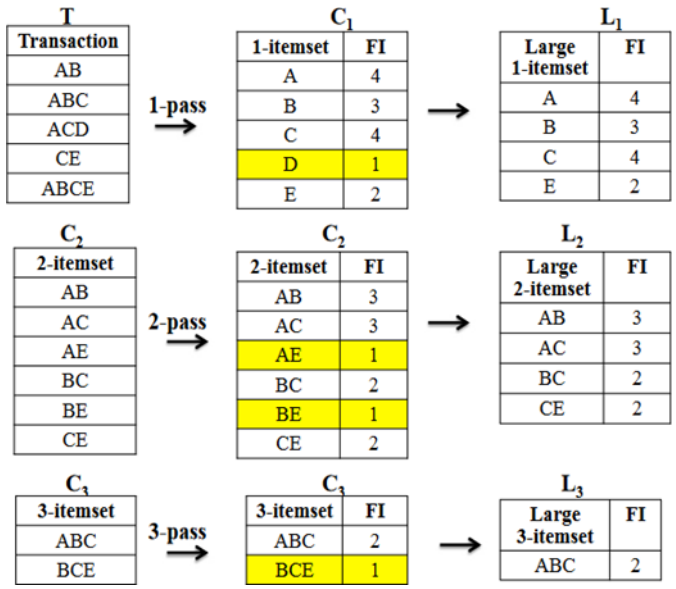
Para poder obtener estas reglas se realizan los siguientes pasos (Lin et al., 2019):

1) Se genera un conjunto de ítems, que se denomina conjunto de ítems candidatos. El algoritmo Apriori se inicia con un único conjunto de ítems (1- itemset).
2) Si el grado de soporte del conjunto de ítems candidatos es mayor o igual que el soporte mínimo, el conjunto de ítems candidatos es un itemset de alta frecuencia (Large 1-itemset). (* Podemos definir un nivel de soporte mínimo para calificar como "frecuente", que depende del contexto. En este caso, soporte mínimo = 2. Por lo tanto, todos son frecuentes.
3) Utilizando la combinación de estos elementos de alta frecuencia, se genera el conjunto de 2 ítems candidatos.
4) Después de obtener el soporte del conjunto de 2 ítems candidatos, se vuelve a encontrar el conjunto de 2 ítems de alta frecuencia, y se utiliza la combinación de estos ítems de 2 ítems de alta frecuencia para generar el conjunto de 3 ítems candidatos.
5) Se compara con el soporte mínimo y se genera un conjunto de ítems de alta frecuencia, que se combina para generar un conjunto de ítems candidatos del siguiente nivel hasta que no se producen nuevos conjuntos de ítems candidatos.

Para una representación gráfica de los procesos realizados por el algoritmo Apriori se tiene la Figura 37.

Figura 37

Ejemplo de uso del algoritmo Apriori



Nota: Extraído de Lin et al. (2019)

De acuerdo con los pasos realizados por el algoritmo Apriori, este realiza una construcción de elementos frecuentes muy costosa debido a que escanea repetidamente la base de datos escaneando cada una de las transacciones para determinar el soporte de cada uno de los conjuntos de ítems candidatos generado, por ello dentro de la literatura se encuentran otros algoritmos como FP-growth.

4.3.3. Algoritmo Fp-Growth

Algoritmo FP-growth (Han et al., 2000) solventa algunas de los problemas de utilizar Apriori en grandes cantidades de transacciones, para ello utiliza el método de divide y vencerás. Este algoritmo se basa en un árbol de patrones frecuentes (FP-Tree) que permite representar de manera compacta y eficiente los patrones frecuentes presentes en el conjunto de datos. FP-Growth es eficiente y escalable para la extracción de patrones frecuentes, ya que la construcción del árbol FP-Tree se realiza en una sola pasada sobre el conjunto de datos, y la extracción de patrones frecuentes se realiza de manera eficiente utilizando el árbol FP-Tree.

El algoritmo FP-Growth consta de dos fases principales:

- Construcción del árbol de patrones frecuentes (FP-Tree): En esta fase, el algoritmo construye un árbol FP-Tree a partir del conjunto de datos de

entrada. Cada transacción del conjunto de datos se recorre y se inserta en el árbol FP-Tree de manera que cada nodo del árbol represente un ítem y su frecuencia en el conjunto de datos. Si un ítem ya existe en el árbol, se incrementa su frecuencia en lugar de crear un nuevo nodo.

- Extracción de patrones frecuentes a partir del árbol FP-Tree: En esta fase, el árbol FP-Tree se utiliza para extraer los patrones frecuentes presentes en el conjunto de datos. Los patrones frecuentes se extraen recursivamente mediante un proceso llamado backtracking. Se comienza en la hoja del árbol y se asciende hacia la raíz, construyendo los patrones frecuentes a medida que se recorre el árbol.

4.3.4. Caso Práctico: Reglas de asociación

El caso práctico se centra en una empresa de retail en línea que desea recomendar productos mediante el Análisis de Cesta de Compras por lo que para ello se realiza la construcción de reglas de asociación que nos permite tener los patrones dentro de las transacciones históricas y poder con ello implementar un sistema recomendador de productos para el sitio web.

Se cuenta con un conjunto de datos de 131 mil registros de ordenes de compras donde se almacenan el id de la orden, el id del producto comprado, su descripción, la cantidad del producto comprado, el cliente, entre otras variables de interés.

```
#####################################################################
# Minería de Reglas de Asociación
#####################################################################
# Importa los paquetes necesarios
library(arules)
library(plyr)
library(RColorBrewer)
library(dplyr)
library(readr)
# Se carga al espacio de trabajo el conjunto de datos de 131 mil
# registros de órdenes del retail online.
```

```
orders_load <- read_csv("orders_A_R.csv")
orders_load
```

```
# A tibble: 1,384,617 × 14
```

	order_id	product_id	add_to_c…¹	reord…²	user_id	order…³	order…⁴
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	49302	1	1	112108	4	4
2	1	11109	2	1	112108	4	4
3	1	10246	3	0	112108	4	4
4	1	49683	4	0	112108	4	4
5	1	43633	5	1	112108	4	4
6	1	13176	6	0	112108	4	4
7	1	47209	7	0	112108	4	4
8	1	22035	8	1	112108	4	4
9	36	39612	1	0	79431	23	6
10	36	19660	2	1	79431	23	6

Dentro de los procesos de preprocesamiento se realiza la selección de las variables que se van a utilizar para la obtención de las reglas y también se convierten los tipos de datos de estas debido que representan una categoría a pesar de estar representadas como números.

```
# Selecciona las variables de interes y convierte las variables a
# tipo factor (categóricas)
orders <- orders_load %>% select(order_id, product_name) %>%
   mutate(order_id=as.factor(order_id)) %>%
   mutate(product_name=as.factor(product_name))
# Muestra el nuevo conjunto de datos preprocesado
head(orders)
```

```
# A tibble: 6 × 2
```

	order_id	product_name
	<fct>	<fct>
1	1	Bulgarian Yogurt
2	1	Organic 4% Milk Fat Whole Milk Cottage Cheese
3	1	Organic Celery Hearts
4	1	Cucumber Kirby

5	1	Lightly Smoked Sardines in Olive Oil
6	1	Bag of Organic Bananas

Para los algoritmos de obtención de reglas de asociación los datos de entrada necesitan presentar ciertos formatos específicos como, por ejemplo:

Tabla 7
Ejemplo de formato de transacciones para algoritmos de reglas de asociación

TID	Elementos
1	pan,leche
2	Pan,pañales,cerveza,huevos
3	Leche,pañales,cerveza,cola
4	Pan,leche,pañales,cerveza
5	Pan,leche,pañales,cola

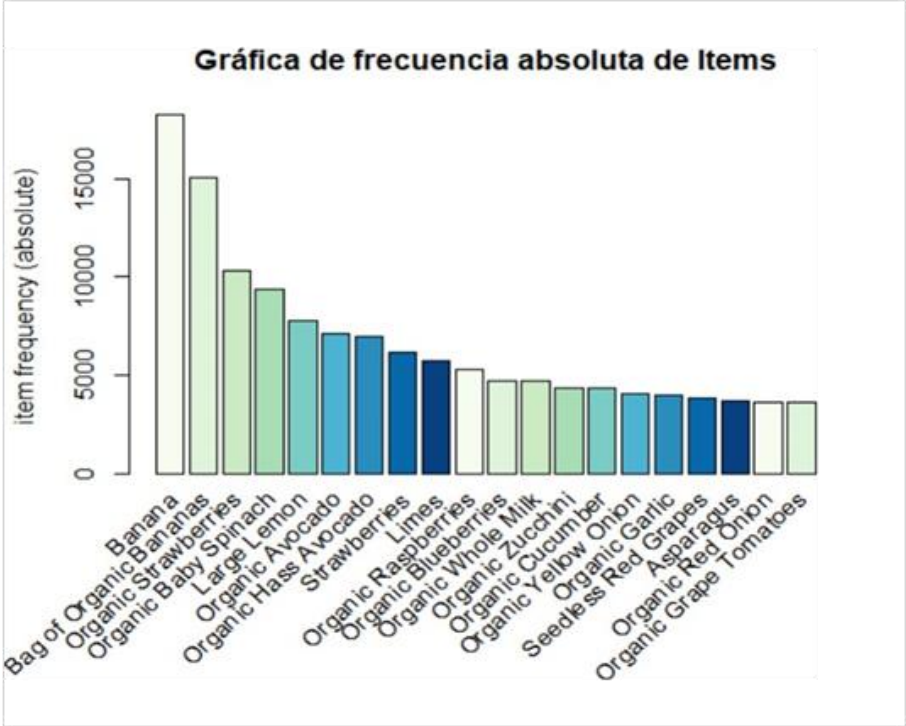
Nota: Autores (2023)

Por ello se realiza una segunda fase de preprocesamiento donde se transforma los registros a un formato compatible con la entrada requerida por el algoritmo de reglas de asociación.

```
# Se realiza la codificación de las compras a un formato de tipo basket
# el cual es admitido para construir un objeto de tipo "transactions"
txns = ddply(orders,c("order_id"),
   function(df1) paste(df1$product_name, collapse = ","))
write.csv(txns$V1,"transactions.csv", quote = FALSE, row.names = FALSE)
txns <- read.transactions('transactions.csv', format ='basket',
      sep=',', header = T)
# Grafica los ítems frecuentes en las transacciones generadas
itemFrequencyPlot(txns, topN=20, type="absolute", col=brewer.pal(10,'Gn Bu'),
      main="Gráfica de frecuencia absoluta de Items")
```

Figura 38

Frecuencia absoluta de ítems en las transacciones



Nota: Autores (2023)

De acuerdo con la Figura 38 se puede visualizar que las bananas, las fresas y las hojas de espinacas jóvenes son los productos que más frecuencia se tiene en las transacciones del retail online.

Una vez analizado los ítems frecuentes se procede a la obtención de las reglas mediante el algoritmo de apriori implementado en R en el paquete **arules.**

```
# Mediante el algoritmo apriori se obtienen las reglas con
# los parametros de soporte y confianza establecidos
AR_rules = arules::apriori(txns, parameter = list(supp = 0.01, conf = 0. 1))
# Se ordenan las reglas para obtener las reglas de mayor
# confianza y se las muestran
rules_conf <- sort (AR_rules, by="confidence", decreasing=TRUE)
inspect(head(rules_conf))
```

lhs		rhs	support	confidence	lift
{Bag}	=>	{Clementines}	0.0104	0.7822	38.19
{Clementines}	=>	{Bag}	0.0104	0.5120	38.19
{Organic Hass Avocado}	=>	{Bag of Organic Bananas}	0.0172	0.3252	2.834
{Organic Raspberries}	=>	{Bag of Organic Bananas}	0.0126	0.3150	2.746
{Organic Avocado}	=>	{Banana}	0.0161	0.2975	2.142
{Strawberries}	=>	{Banana}	0.0139	0.2944	2.120

Con las reglas obtenidas se puede desarrollar algunas estrategias para el retail como la de recomendación de productos o también hacer cambios en las estanterías físicas para un despacho más eficiente de los pedidos. Colocando los aguacates en las mismas estanterías de las bananas o en la web por ejemplo recomendando que compren bananas a los clientes que en su carrito de compras ya tengan frambuesas o fresas.

Con los parámetros de soporte y de confianza se pueden refinar las reglas a obtener con el algoritmo.

4.4. Reducción de dimensionalidad

4.4.1. Definiciones

La reducción de la dimensionalidad es la tarea de reducir la dimensionalidad de los patrones, mientras se conserva la información importante (Kramer, 2013). Usualmente se usan para mejorar los resultados de los análisis supervisados como por ejemplo la clasificación o regresión. Esta técnica es importante en la minería de datos ya que a menudo se trabaja con conjuntos de datos de alta dimensionalidad, lo que puede llevar a problemas de escalabilidad, ruido, redundancia y sobreajuste.

Dos de los principales enfoques para la reducción de dimensionalidad es la selección de características y la extracción de características. La selección de características implica seleccionar un subconjunto de variables relevantes del conjunto de datos original y la extracción de características implica la creación de nuevas variables a partir de las variables originales, que capturan la información más relevante del conjunto de datos.

4.4.2. Análisis de componentes principales

El Análisis de componentes principales (PCA por sus siglas en inglés) es una de las técnicas no supervisadas de reducción de la dimensionalidad más utilizadas (Kumar & Pande, 2022), esta usa el enfoque de extracción de características. El principio fundamental del PCA es mitigar las características de entrada para mejorar el rendimiento de los clasificadores predictivos de ML. Sin embargo, PCA asume la relación lineal entre todas las variables y no funciona bien si encuentra relaciones no lineales. Para aplicar un análisis de componentes principales es importante que los datos estén normalizados antes de su procesamiento, esto evita que los atributos con dominios grandes dominen a atributos con dominios pequeños.

PCA calcula los vectores unitarios ortogonales que proporcionan una base para los datos de entrada normalizados. Estos vectores son perpendiculares entre sí y toman el nombre de *Componentes Principales*. Estos componentes se clasifican y ordenan de forma decreciente en base a su importancia o fuerza. Por lo tanto, el primer componente es aquel que muestra la mayor variación entre los datos y el segundo componente muestra la siguiente varianza más alta y así sucesivamente.

4.4.3. Caso Práctico: Análisis de componentes principales (PCA)

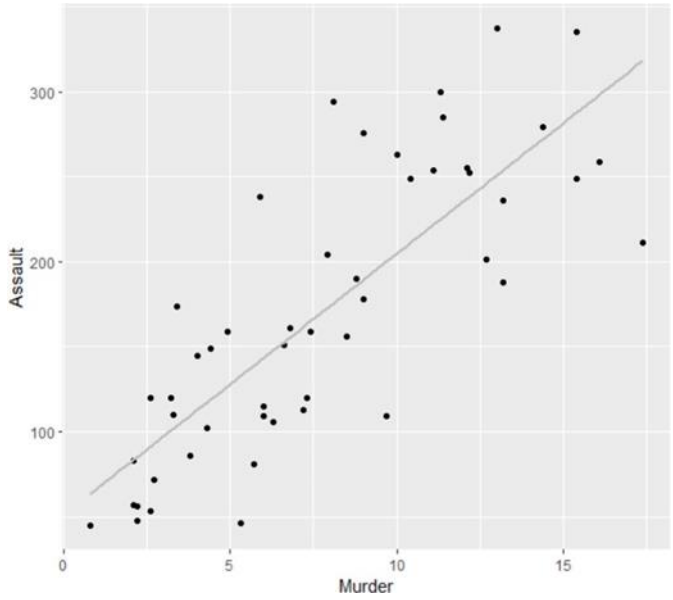
Se utilizará el conjunto de datos USArrests el cual contiene estadísticas, en arrestos por cada 100,000 residentes por agresión, asesinato y violación en cada uno de los 50 estados de EE. UU. en 1973, a los cuales en esta ocasión se realizará un análisis de componentes principales, para lo cual se analizará a breve rasgos la linealidad de los datos debido a que PCA asume la relación lineal entre las variables.

```
#####################################################################
# Análisis de Componentes Principales
#####################################################################
# Se carga y se visualiza los datos
data("USArrests")
# Verificamos gráficamente la linealidad en los datos
```

```
ggplot(data = USArrests, aes(x=Murder, y=Assault))+
   geom_point()+
   geom_smooth(method = 'lm', se = F, color='gray', linetype='solid')
```

Figura 39

Gráfica de la variable Murder en función de variable Assault



Nota: Autores (2023)

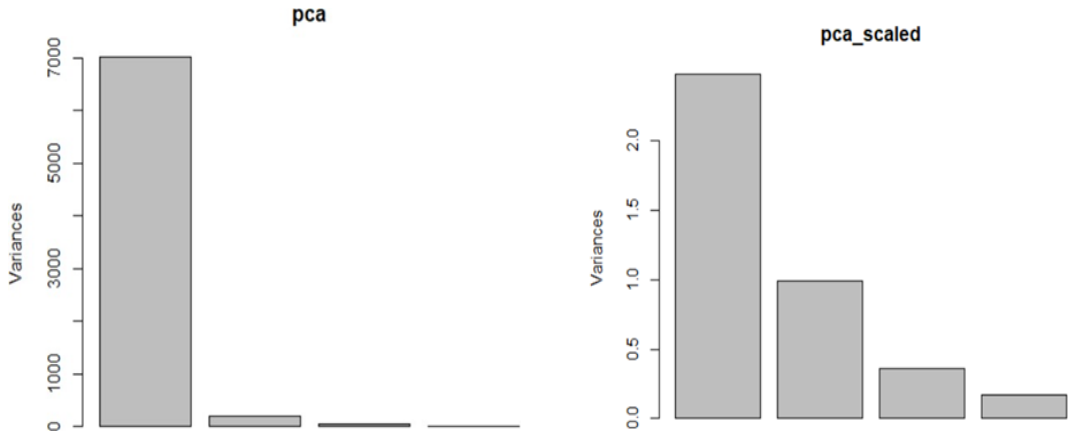
Una vez cargados los datos y mostrado la gráfica, se puede visualizar en la Figura 39 la relación lineal positiva que tienen las variables expuestas. Por lo que se procede con el análisis de componentes principales para realizar una reducción de dimensionalidad mediante la extracción de características. Antes de ello, a manera de ejemplo se muestra las varianzas de los componentes encontrados cuando se aplica una normalización y cuando no.

```
#####################################################################
# Ejecuta el algoritmo de PCA con el conjunto de datos seleccionado
pca <- prcomp(USArrests)
# Ejecuta el algoritmo de PCA con el conjunto de datos seleccionado
# pero aplica un escalamiento en los datos.
pca_scaled <- prcomp(USArrests, scale. = TRUE)
# Gráfica de las varianzas de los componentes obtenidos
plot(pca)
# Gráfica de las varianzas de los componentes obtenidos
```

```
# cuando se realiza el escalado de los datos
plot(pca_scaled)
```

Figura 40

Graficas de las varianzas de un PCA con datos sin escalar versus un PCA con datos escalados



Nota: Autores (2023)

En las gráficas anteriores se puede observar las diferencias en las varianzas de los Componentes Principales cuando no se realiza un escaldado y cuando se realiza un escalado. Estos resultados corroboran la teoría que debe de escalarse los atributos antes de un análisis de PCA.

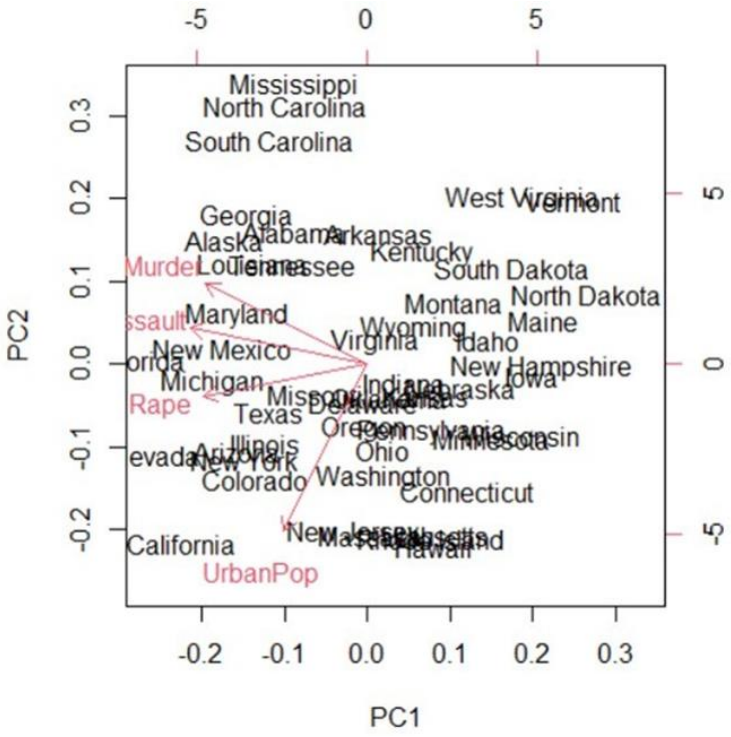
```
# Muestra un resumen del resultado del algoritmo de PCA
> summary(pca)
Importance of components:
                          PC1      PC2      PC3      PC4
Standard deviation     83.7324 14.21240   6.4894   2.4879
Proportion of Variance  0.9655  0.02782   0.0058  0.00085
Cumulative Proportion   0.9655  0.99335   0.9991  1.00000
> summary(pca scaled)
Importance of components:
                          PC1      PC2      PC3      PC4
Standard deviation      1.5749   0.9949  0.59713  0.41645
Proportion of Variance  0.6201   0.2474  0.08914  0.04336
Cumulative Proportion   0.6201   0.8675  0.95664  1.00000
```

De acuerdo con los resultados de la proporción de la varianza de los componentes PC1 y PC2 se pueden extraer la mayor cantidad de información relevante a partir de los 4 atributos de entrada.

```
# Gráfica del primer y segundo componente obtenido
biplot(pca_scaled)
# Calcula los nuevos valores en base a los componentes encontrados
pc1 <- apply(pca_scaled$rotation[,1] * USArrests, 1, sum)
pc2 <- apply(pca_scaled$rotation[,2] * USArrests, 1, sum)
# Agrega los valores calculados con los componentes
USArrests$pc1 = pc1
USArrests$pc2 = pc2
USArrests[, c(5,6)]
```

En la siguiente gráfica se visualiza los estados proyectados sobre los dos Componentes Principales seleccionados y adicionalmente se muestra los valores en ambos componentes de cada uno de los estados.

Figura 41
Gráfica de los Componentes Principales PC1 y PC2



Nota: Autores (2023)

```
> USArrests[,c(5,6)]
                pc1          pc2
Alabama     -20.78407  -264.100705
Alaska     -112.95614   -68.905147
Arizona    -157.32936   188.736847
Arkansas    -47.12650    62.490457
California  -28.88522  -322.518448
Colorado    -91.15214   -53.740141
Connecticut -72.54273    26.655060
```

```
USArrests[,c(5,6)]
                pc1          pc2
Alabama     -20.78407  -264.100705
Alaska     -112.95614   -68.905147
Arizona    -157.32936   188.736847
Arkansas    -47.12650    62.490457
California  -28.88522  -322.518448
Colorado    -91.15214   -53.740141
Connecticut -72.54273    26.655060
```

GLOSARIO

Glosario

5.1. Glosario de siglas

ANN	Artificial Neural Network (Red Neuronal Artificial)
ASCII	American Standard Code for Information Interchange (Código Estadounidense Estándar para el Intercambio de Información)
BN	Bayesian Networks (Redes Bayesianas)
CDSS	Clinical Decision Support System (Sistemas de soporte a las decisiones clínicos)
CHD	Coronary Heart Disease (Enfermedades coronarias del corazón) CPU
CPU	Central Processing Unit (Unidad central de procesamiento)
CRAN	Comprehensive R Archive Network (Red integral de archivos R)
CRISP-DM	Cross Industry Standard Process for Data Mining (Proceso Estándar Inter-Industrias para Minería de Datos)
DBMS	DataBase Management System (Sistema de gestión de bases de datos)
DM	Data Mining (Minería de Datos)
EM	Expectation–maximization (Esperanza-Maximización)
FP-Tree	Frequent Pattern Tree (Árbol de patrones frecuentes)
IA	Inteligencia Artificial
IBM	International Business Machines
KDD	Knowledge Discovery in Database (Descubrimiento de conocimiento en bases de datos)
KEEL	Knowledge Extraction based on Evolutionary Learning (Extracción de conocimiento basada en el aprendizaje evolutivo)
KNN	K nearest neighbors (K vecinos más cercanos)
LDA	Latent Dirichlet Allocation (Asignación Latente de Dirichlet)
LRT	Likelihood Ratio Test (Prueba de razón de verosimilitud)
MAR	Missing at Random (Valores faltantes al azar)

MARS	Multivariate Adaptive Regression Spline (Regresión spline adaptativa multivariante)
MCAR	Missing Completely at Random (Valores faltantes completamente al azar)
MDI	Median imputation (Imputación usando la mediana)
MDLP	Minimum Description Length Principle (Principio de Longitud de Descripción Mínima)
MI	Mean imputation (Imputación usando la media)
ML	Machine Learning (Aprendizaje automático)
MLP	Multilayer Perceptron (Perceptrón Multicapa)
MSE	Mean Squared Error (Error Cuadrático Medio)
NB	Naïve Bayes (Ingenuo Bayes)
NMAR	Missing Not at Random (Valores faltantes no al azar o no ignorables)
PCA	Principal Component Analysis (Análisis de componentes principales)
RBF	Radial Basis Function (Función de base radial)
RDBMS	Relational Database Management System (Sistema de Gestión de Bases de Datos Relacionales)
RNA	Red Neuronal Artificial
ROC	Receiver Operating Characteristics (Característica operativa del receptor)
SAS	Statistical Analysis Systems (Sistemas de análisis estadístico)
SEMMA	Sample, Explore, Modify, Model, Assess (Muestreo, Explorar, Modificar, Modelar y Evaluar)
SQL	Structure Query Language (Lenguaje de consultas estructuradas)
SVM	Support Vector Machine (Máquinas de vectores de soporte)
TID	Transaction Identifier (Identificador de transacción)
TS	Técnicas Supervisadas
VI	Variable independiente
VIs	Variables independientes

5.2. Glosario de términos

1R	Es un método de discretización supervisado que utiliza el binning.
AIC	El Criterio de información de Akaike es una medida de la calidad relativa de un modelo estadístico.
Backtracking	Es una estrategia para encontrar soluciones a problemas que satisfacen restricciones.
C	Es un poderoso lenguaje de programación de propósito general.
C4.5	Algoritmo usado para generar un árbol de decisión desarrollado por Ross Quinlan.
Data Frame	Estructuras de datos de dos dimensiones (organizados en columnas y filas) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas.
Datamining	Minería de datos.
GNU	Sistema operativo de tipo UNIX.
Gráfica normal Q-Q	Este gráfico es un gráfico entre "Cuantiles teóricos" y "Valores ordenados"
GUI	Interfaces gráficas de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.
IB1	Algoritmo basado en técnicas de vecinos más cercanos.
IDE	Aplicación de software que ayuda a los programadores a desarrollar código de software de manera eficiente.
KEEL	Herramienta de software Java de código abierto que se puede utilizar para una gran cantidad de diferentes tareas de descubrimiento de datos de conocimiento.
Lisp	Es un lenguaje de programación funcional que fue diseñado para una fácil manipulación de cadenas de datos.
NA	En R se usa para representar cualquier valor 'no disponible' o valores faltantes.
Outliers	Son datos anormales dentro de un conjunto de datos.

Pentabytes	Unidad de medida de la capacidad de memoria o del tamaño de los datos equivalente a 1024 gigabytes.
R	Entorno y lenguaje de programación con un enfoque al análisis estadístico.
R2	También llamado coeficiente de determinación, mide cómo se ajusta un modelo de regresión a los datos reales.
Retail	Es un tipo de comercio que se caracteriza por vender al por menor.
S	Lenguaje de programación estadístico.
SPSS	Programa estadístico informático que originalmente se usaba únicamente en las investigaciones de las ciencias sociales y en las ciencias aplicadas
UCI	Es la University of California, Irvine
UNIX	Sistema operativo portable, multitarea y multiusuario.

Referencias Bibliográficas

Referencias Bibliográficas

Acuna, E., & members of the CASTLE group at UPR-Mayaguez Puerto Rico. (2015). *dprep: Data Preprocessing and Visualization Functions for Classification*. R Package version 3.0.2.

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. *SIGMOD Rec.*, *22*(2), 207–216.

Akaike, H., Petrov, B. N., & Csaki, F. (1973). *Second international symposium on information theory*. Akademia Kiado.

Al Shalabi, L., Shaaban, Z., & Kasasbeh, B. (2006). Data mining: A preprocessing engine. *Journal of Computer Science*, *2*(9), 735–739.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*(2–3), 255–287.

Alcalá-Fdez, J., Sánchez, L., García, S., Jesús, M. D., Ventura, S., Guiu, J. M., Otero, J., Romero, C., Bacardit, J., Santos, V., Fernández, J. C., & Herrera, F. (2009). KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, *13*, 307–318.

Amraoui, H., Mhamdi, F., & Elloumi, M. (2019). Fast Bat Algorithm for Predicting Diabetes Mellitus Using Association Rule Mining. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11888 LNAI*.

Ara Shaikh, A., Nirmal Doss, A., Subramanian, M., Jain, V., Naved, M., & Khaja Mohiddin, M. (2022). Major applications of data mining in medical. *Materials Today: Proceedings*, *56*, 2300–2304.

Bachman, C. W. (1972). The Evolution of Storage Structures. *Communications of the ACM*, *15*(7), 628–634.

Batayev, N. (2018). Gas Turbine Fault Classification Based on Machine Learning Supervised Techniques. *2018 14th International Conference on Electronics Computer and Computation (ICECCO)*, 206–212.

Bell, L., Chambers, J. M., Bickel, P. J., Cleveland, W. S., & Dudley, R. M. (1984). *S an Interactive Environment for Data Analysis and Graphics*. CRC Press, Inc.

Boriah, S., Chandola, V., & Kumar, V. (2008). Similarity Measures for Categorical Data: A Comparative Evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)* (pp. 243–254).

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152.

Bramer, M. (2016). Principles of datamining. In *Principles of datamining* (3rd ed.). Springer-Verlag.

Buyanova, S. N., Shchukina, N. A., Temlyakov, A. Y., & Glebov, T. A. (2023). Artificial intelligence in pregnancy prediction. *Russian Bulletin of Obstetrician- Gynecologist*, *23*(2), 83 – 87.

Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, *557*, 317–331.

Chakraborty, S., Jana, G. C., Kumari, D., & Swetapadma, A. (2020). An improved method using supervised learning technique for diabetic retinopathy detection. *International Journal of Information Technology*, *12*(2), 473–477.

Chen, M., Yin, C. J., & Xi, Y. P. (2011). A New Clustering Algorithm Partition K-Means. *Advanced Materials and Computer Science*, *474*, 577–580.

Chen, X. (2020). Analysis of Classification of Discretization Method. *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 186–190.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387.

Colonna, L. (2013). A taxonomy and classification of data mining. *SMU Sci. \& Tech. L. Rev.*, *16*, 309.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*.

Di Noia, A., Martino, A., Montanari, P., & Rizzi, A. (2020). Supervised machine learning techniques and genetic optimization for occupational diseases risk prediction. *Soft Computing*, *24*(6), 4393–4406.

Dua, D., & Graff, C. (2017). *{UCI} Machine Learning Repository*.

Ebrahim, O. A., & Derbew, G. (2023). Application of supervised machine learning algorithms for classification and prediction of type-2 diabetes disease status in Afar regional state, Northeastern Ethiopia 2021. *Scientific Reports*, *13*(1).

Eskin, E., Prerau, A., & Sal, P. S. (2002). A Geometric Framework for Unsupervised Anomaly Detection. In B. D. Sushil & Jajodia (Eds.), *Applications of Data Mining in Computer Security* (pp. 77–101). Springer US.

Evaluation for Decision Support Systems. *Algorithms*, *15*(4).

Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). Cluster analysis: Fifth edition. In *Cluster Analysis: Fifth Edition*.

Fayyad, U. M., & Irani, K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes. In *Conference on Artificial Intelligence* (pp. 1022–1027).

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, *17*(3), 37–54.

Ge, Y., Li, Z., & Zhang, J. (2023). A simulation study on missing data imputation for dichotomous variables using statistical and machine learning methods. *Scientific Reports*, *13*(1).

Giha, F. E., Singh, Y. P., & Ewe, H. T. (2003). Customer profiling and segmentation based on association rule mining technique. *Proceedings of the IASTED International Conference on Software Engineering and Applications*, *7*, 37–42.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. Gordan, M., Sabbagh-Yazdi, S.-R., Ismail, Z., Ghaedi, K., Carroll, P., McCrum, D., &

Gupta, V. K., & Parsad, R. (n.d.). *History of Statistics on Timeline*.

Hacibeyoglu, M., & Ibrahim, M. H. (2018). EF_Unique: An Improved Version of Unsupervised Equal Frequency Discretization Method. *Arabian Journal for Science and Engineering*, *43*(12), 7695–7704.

Hamzehi, M., & Hosseini, S. (2022). Business intelligence using machine learning algorithms. *Multimedia Tools and Applications*, *81*(23), 33233 – 33251.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *SIGMOD Rec.*, *29*(2), 1–12.

Happy, S. L., Dantcheva, A., & Bremond, F. (2019). A Weakly Supervised learning technique for classifying facial expressions. *Pattern Recognition Letters*, *128*, 162–168.

Heinze, G., Wallisch, C., & Dunkler, D. (2018). Variable selection - A review and recommendations for the practicing statistician. *Biometrical Journal. Biometrische Zeitschrift*, *60*(3), 431–449.

Hernández Millán, Á. R., Mendoza-Moreno, M., Portocarrero López, L. M., & Castro-Romero, A. (2018). Comparative Study of Machine Learning Supervised Techniques for Image Classification Using an Institutional

Identification Documents Dataset. *2018 Congreso Internacional de Innovación y Tendencias En Ingeniería (CONIITI)*, 1–6.

Jahangiri, M., Kazemnejad, A., Goldfeld, K. S., Daneshpour, M. S., Mostafaei, S., Khalili, D., Moghadas, M. R., & Akbarzadeh, M. (2023). A wide range of missing imputation approaches in longitudinal data: a simulation study and real data analysis. *BMC Medical Research Methodology*, *23*(1).

Jaramillo, I. F., Garzás, J., & Redchuk, A. (2021). Numerical Association Rule Mining from a Defined Schema Using the VMO Algorithm. *Applied Sciences*, *11*(13).

Jaramillo, I. F., Villarroel-Molina, R., Pico, B. R., & Redchuk, A. (2021). A Comparative Study of Classifier Algorithms for Recommendation of Banking Products. *Trends and Applications in Information Systems and Technologies: Volume 2 9*, 253–263.

Jiang, S., Li, X., Zheng, Q., & Wang, L. (2009). Approximate equal frequency discretization method. *2009 WRI Global Congress on Intelligent Systems*, *3*, 514–518.

Jiawei, H., Jian, P., & Hanghang, T. (2022). *Data Mining: Concepts and Techniques.* (4th ed.). Morgan Kaufmann.

Jiawei, H., Kamber, M., & Pei, J. (2012). *Data mining: concepts and techniques.* (3rd ed., Vol. 3). Morgan Kaufmann.

KEEL. (s.f.). *Unsupervised data sets*. KEEL. https://sci2s.ugr.es/keel/category.php?cat=uns

Kerber, R. (1992). Chimerge: discretization of numeric attributes. *Proceedings Tenth National Conference on Artificial Intelligence*, 123–128.

Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization Techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, *32*(1), 47–58.

Kramer, O. (2013). Dimensionality Reduction. In *Dimensionality Reduction with Unsupervised Nearest Neighbors* (pp. 33–52). Springer Berlin Heidelberg.

Kumar, K., & Pande, B. P. (2022). 2 - Applications of supervised machine learning techniques with the goal of medical analysis and prediction: A case study of breast cancer. In S. Roy, L. M. Goyal, V. E. Balas, B. Agarwal, & M. Mittal (Eds.), *Predictive Modeling in Biomedical Data Mining and Analysis* (pp. 21– 47). Academic Press.

Kutuzova, T., & Melnik, M. (2018). Market basket analysis of heterogeneous data sources for recommendation system improvement. *Procedia Computer Science*, *136*, 246–254.

Li, K.-P., & Porter, J. E. (1988). Normalizations and selection of speech segments for speaker recognition scoring. *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, 595–596.

Li, Z., Lin, X., Zhang, Q., & Liu, H. (2020). Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, *56*, 100694.

Lin, H.-K., Hsieh, C.-H., Wei, N.-C., & Peng, Y.-C. (2019). Association rules mining in R for product performance management in industry 4.0. *Procedia CIRP*, *83*, 699–704.

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, *6*(4), 393–423.

Liu, J. W. Q. L. S. W. Y. (2019). Sentiment Analysis Method Based on Kmeans and Online Transfer Learning. *Computers, Materials & Continua*, *60*(3), 1207– 1222.

Lou, N. (2022). Analysis of the Intelligent Tourism Route Planning Scheme Based on the Cluster Analysis Algorithm. *Computational Intelligence and Neuroscience*, *2022*.

Majid, A. M., & Utomo, W. H. (2021). Application of discretization and adaboost method to improve accuracy of classification algorithms in predicting diabetes mellitus. *ICIC Express Letters, Part B: Applications*, *12*(12), 1177 – 1184.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

McNeil, D. R. (1977). *Interactive Data Analysis*. New York: Wiley.

Mondal, P. K., Foysal, K. H., Norman, B. A., & Gittner, L. S. (2023). Predicting Childhood Obesity Based on Single and Multiple Well-Child Visit Data Using Machine Learning Classifiers. *Sensors*, *23*(2).

Moret, B. M. E. (1982). Decision Trees and Diagrams. *ACM Comput. Surv.*, *14*(4), 593–623.

Müllner, D. (2011). *Modern hierarchical, agglomerative clustering algorithms*. Odhiambo, P., Okello, H., Wakaanya, A., Wekesa, C., & Okoth, P. (2023).

Mutational signatures for breast cancer diagnosis using artificial intelligence. *Journal of the Egyptian National Cancer Institute*, *35*(1).

Park, J., Müller, J., Arora, B., Faybishenko, B., Pastorello, G., Varadharajan, C., Sahu, R., & Agarwal, D. (2023). Long-term missing value imputation for time series data using deep neural networks. *Neural Computing and Applications*, *35*(12), 9071 – 9091.

Pérez, C., & Santín, D. (2008). Minería de datos. Técnicas y herramientas. Madrid: Paraninfo. *Revista Lasallista de Investigación-Árboles de Decisión Como Metodología Para Determinar El Rendimiento Académico En*, *104*.

Queen, J. Mac. (1967). Some Methods for Classification and Analysis of Multivariate Obser- vations. *Roceedings of the 5 Th Berkeley Symposium on Mathematical Statistics and Proba- Bility*, 281–297.

Ramírez, C., Orallo, J. H., & Quintana, J. R. (2004). *Introducción a la Minería de Datos* (1st ed.). Pearson.

Rezayi, S., Maghooli, K., & Saeedi, S. (2021). Applying Data Mining Approaches for Chronic Kidney Disease Diagnosis. *International Journal of Intelligent Systems and Applications in Engineering*, *9*(4), 198–204.

Russell, S. J., & Norvig, P. (1995). Learning in Neural and Belief Networks. In I. Prentice-Hall (Ed.), *Artificial Intelligence A Modern Approach*. Alan Apt.

Rwzhang. (2018). *seeds dataset* [Data set]. https://www.kaggle.com/datasets/rwzhang/seeds-dataset

Samali, B. (2022). State-of-the-art review on advancements of data mining in structural health monitoring. *Measurement*, *193*, 110939.

SAS Institute Inc. All. (2015). SAS Enterprise miner: Reveal valuable insights with powerful data mining software. http://www.sas.com/en_us/software/analytics/enterprise-miner.html.

Shanwu, S., Shuru, T., & Nan, W. (2020). Study on The Prediction of Electricity Stealing Based on Improved SMOTE Algorithm and Ensemble Learning. *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 242–248.

Shmueli, G. (2010). To explain or to predict? *Statistical Science*, *25*(3), 289–310. Silva, H., & Bernardino, J. (2022). Machine Learning Algorithms: An Experimental

Sokolova, M., & Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manage.*, *45*(4), 427–437.

Šulc, Z., & Řezanková, H. (2019). Comparison of Similarity Measures for Categorical Data in Hierarchical Clustering. *Journal of Classification*, *36*(1), 58–72.

Sun, X., Yin, Y., Yang, Q., & Huo, T. (2023). Artificial intelligence in cardiovascular diseases: diagnostic and therapeutic perspectives. *European Journal of Medical Research*, *28*(1).

Wahyuni, S. N., Khanom, N. N., & Astuti, Y. (2023). K-Means Algorithm Analysis for Election Cluster Prediction. *International Journal on Informatics Visualization*, *7*(1), 1–6.

Wikipedia contributors. (2023, diciembre 3). *Lisp (programming language)*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Lisp_(programming_language)&oldid=1188060921

Williams, G. (2012). *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, *1*, 29–40.

Zwolenski, M., & Weatherill, L. (2014). *The Digital Universe Rich Data and the Increasing Value of the Internet of Things*. Australian Journal of Telecommunications and the Digital Economy.

RESUMEN

Este libro trata sobre conceptos elementales junto con scripts cortos de código basado en R Project para hacer análisis inteligente de datos. La relación entre la teoría y la práctica es fundamental en la comprensión de una disciplina, así la aplicación de procedimientos y funciones específicas en tareas elementales es el propósito de este texto. La idea central del texto tiene origen en la asignatura denominada “Análisis Inteligente de Datos”, una cátedra en la que el profesor aporta con elementos fundamentales basados en conceptos y ejercicios prácticos usando R Project. Hoy en día, la disponibilidad de herramientas para la minería de datos es sin duda muy grande. Usuarios con conocimientos básicos pueden aprovechar de utilitarios intuitivos implementados en poderosos entornos de desarrollo. Nosotros hemos querido dar un enfoque al texto hacia una audiencia con mayor relación a la programación y software. Específicamente que constituya una guía básica para estudiantes que inician en el campo de la Inteligencia de datos.

Palabras Clave: Inteligencia, Datos, R project.

http://www.editorialgrupo-aea.com

Editorial Grupo AeA

editorialgrupoaea

Editorial Grupo AEA

ISBN: 978-9942-651-20-4