Método para generación y ordenación de reglas de clasificación. Diseño y estudio computacional. Aplicación a la Inteligencia de Negocio

Memoria presentada por
Jesús Javier Rodríguez Sala

Para optar al grado de Doctor
por la Universidad Miguel Hernández de Elche,
dirigida por:

Dr. Alejandro Rabasa Dolado
y
Dr. José María Gómez Gras

Septiembre 2014
Departamento de Estadística, Matemáticas e Informática

UNIVERSITAS
Miguel
Hernández

D. Alejandro Rabasa Dolado, Contratado Doctor del Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche, y D. José María Gómez Gras, Catedrático de Organización de Empresas de la Universidad Miguel Hernández de Elche

CERTIFICAN que la presente Memoria "**Método para generación y ordenación de reglas de clasificación. Diseño y estudio computacional. Aplicación a la Inteligencia de Negocio**", ha sido realizada bajo su dirección en el Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche, por D. Jesús Javier Rodríguez Sala, y que constituye su tesis para optar al grado de Doctor por la Universidad Miguel Hernández de Elche.

Y para que conste, en cumplimiento de la legislación vigente, firman el presente certificado

En Elche, a 24 de Septiembre de 2014

Fdo: Alejandro Rabasa Dolado

Fdo: José María Gómez Gras

UNIVERSITAS
Miguel
Hernández

D. José María Amigó García, Catedrático de Matemática Aplicada y Director del Departamento de Estadística, Matemáticas e Informática de la Universidad Miguel Hernández de Elche,

INFORMA: Que da su conformidad para la defensa pública de la Tesis Doctoral presentada por D. Jesús Javier Rodríguez Sala, que lleva por título "**Método para generación y ordenación de reglas de clasificación. Diseño y estudio computacional. Aplicación a la Inteligencia de Negocio**",

Y para que conste a los efectos oportunos, emite el siguiente informe

En Elche, a 24 de Septiembre de 2014

Fdo: José María Amigó García

UNIVERSITAS
Miguel
Hernández

A mi familia

UNIVERSITAS
Miguel
Hernández

AGRADECIMIENTOS

El trabajo que se presenta en esta memoria ha sido posible porque un buen día uno entra a formar parte de un grupo en el que “todo conspira” para que, tarde o temprano (tarde en mi caso), ciertas cosas ocurran sí o sí.

A lo largo de mi vida como PDI de la Universidad Miguel Hernández, han sido muchos los compañeros, y aquí incluyo también al personal de administración, que de una u otra manera hacen que tu experiencia sea grata y enriquecedora. Muchas gracias a todos.

En un grano más fino, gracias a los compañeros de departamento con los que el contacto más cotidiano me han ayudado a hacer mi trabajo en todos estos años, gracias por las valiosas lecciones que he aprendido junto a vosotros y por haber sido fuente de inspiración y motivación.

También quiero agradecer a la empresa MUSTANG, especialmente a Tomas Ródenas, Ezequiel Sánchez y Francisco Belda, por el tiempo que me han dedicado para poder realizar una parte importantes de mi investigación.

Refinando el grano hasta el nivel “atómico”, me toca hablar de los compañeros más cercanos, con los que más estrechamente uno ha trabajado (y espero seguir trabajando).

Gracias Rafa, por tener a FENIX siempre a punto y por hacerme descubrir las bondades de la carne a la piedra. Gracias Laureano, buen amigo, proveedor de datos y excelente programador, uno de los mejores que ha visto esta casa. Gracias Agustín y Laura, mis becarios favoritos.

Y, naturalmente, expresar mi agradecimiento a mis directores de tesis, empezando por Alejandro, mi compañero y mi amigo, gracias por estos catorce años de sincera amistad y camaradería. Lo que son las cosas, empezamos con Joe Cocker en el cuarto del fax y la fotocopiadora y ahora tenemos una empresa. Gracias Alex por empeñarte en que fuera doctor (te has ganado de sobra cada almendrado que tu familia y tú os habéis comido, tranquilo que ya queda poco para Navidad).

Y, gracias José María, también, por embarcarte en esta empresa (y en la otra) de codirigir este trabajo cuando aún no conocías mucho a este doctorando, por tu empuje y por la importante labor que has desempeñado, sin tu esfuerzo esta tesis aún no sería una realidad.

… y recordad:
somos buenos
;-)

UNIVERSITAS
Miguel
Hernández

ÍNDICE

UNIVERSITAS
Miguel
Hernández

ALGORITMOS

UNIVERSITAS
Miguel
Hernández

TABLAS

FIGURAS

UNIVERSITAS
Miguel
Hernández

RESUMEN

Las reglas de clasificación son un método clásico para la construcción de modelos que reproducen el comportamiento de cierto tipo de sistemas a partir de conjuntos de datos procedentes de dichos entornos. Estos modelos, con frecuencia, están formados por un número muy elevado de reglas, lo que los hace difíciles de gestionar por parte de analistas que pretendan comprenderlos y poder tomar decisiones sobre los sistemas modelados.

En el campo de la Minería de Datos son bien conocidos los métodos de generación de reglas de clasificación (ID3, C4.5, CART, etc...) así como los mecanismos para reducción de dichos conjuntos de reglas (pre-poda o post-poda) para generar modelos más manejables. Todos estos métodos han funcionado correctamente para resolver muy diversos tipos de problemas, pero en el ámbito de la toma de decisiones estratégicas en los negocios, la tendencia actual es poder disponer de información útil con la mayor rapidez para que el proceso de tomas de decisiones sea también lo más ágil posible. Además, con la cada vez mayor implantación de nuevas fuentes de información como son los portales de comercio electrónico, o las redes sociales, el caudal de datos a considerar es cada vez mayor. Esta nueva problemática hace necesaria la búsqueda de nuevos métodos que, sin pérdida de eficacia en los análisis resultantes, mejoren la eficiencia de los mismos. En otras palabras, se requiere de nuevos mecanismos capaces de proporcionar modelos de calidad y además, que lo hagan en menos tiempo.

En el presente trabajo se presenta un nuevo método para la extracción de reglas de clasificación, al que se ha denominado CREA ('*Classification Rules Extraction Algorithm*'). Este método integra en un único procedimiento la tarea de generación de reglas de clasificación siguiendo el criterio de ganancia de información de ID3 (Quinlan 1979), junto con el método RBS (Rabasa 2009) de reducción de sistemas de reglas por regiones de significancia (post-poda). El método desarrollado no es una mera conexión en serie de los procesos ID3 y RBS, sino que para su implementación se han diseñado ciertas estructuras de datos y algoritmos con objeto de conseguir mejorar la eficiencia con respecto a otras implementaciones de métodos análogos.

La presente memoria incluye un estudio exhaustivo de la complejidad computacional del método CREA, tanto a nivel teórico como a nivel empírico, este segundo, basado en la ejecución del método con 450 conjuntos de datos simulados que han permitido estudiar la evolución del tiempo de ejecución del método bajo diferentes condiciones de carga. Se ha comprobado como el número de ejemplos de un conjunto de datos ('N'), el número de atributos o columnas de los mismos ('C') y el número de diferentes valores que estos atributos pueden tomar ('V') afectan de diferente manera en los tiempos de procesamiento observados, siendo estas observaciones compatibles con el estudio teórico realizado.

Para comprobar, no sólo la eficiencia del método, sino también su eficacia, se ha probado con un conjunto de datos empresariales proporcionados por la empresa MTNG Global Experience S.L. (MUSTANG) para generar modelos de reglas cuya validez ha sido corroborada por expertos de la propia empresa.

Por último, cabe señalar que la presente investigación ha sido financiada en parte por el proyecto precompetitivo Bancaja-UMH 2011, lo cual ha permitido difundir los resultados del mismo en diversos congresos nacionales e internacionales: XXXIV Congreso Nacional de Estadística e Investigación Operativa (SEIO septiembre 2013), y DATA 2013 (Wessex Intitute of Technology, WIT).

Capítulo 1
Estado del arte

1.1. Introducción a las técnicas de Data Mining

A medida que las organizaciones van recopilando más y más información fruto de las operaciones que realizan cotidianamente (y en gran medida, a causa de la incorporación de Internet en sus modelos de negocio), el proceso de extraer información relevante empieza a necesitar de automatismos que preparen, manipulen y sondeen los datos, y extraigan de ellos la información más interesante en cada caso. Hallar soluciones a esta problemática es la razón de ser de la Minería de Datos (Data Mining), una disciplina entre la Estadística y la Informática que proporciona una serie de métodos analíticos orientados para realizar dichas tareas prospectivas sobre grandes conjuntos de datos.

Los análisis de datos se pueden realizar con dos finalidades diferentes (véase tabla 1.1): finalidad descriptiva, con intención de mostrar el comportamiento que tienen ciertas variables hasta el instante actual; y finalidad predictiva, con el objetivo de anticiparse a la evolución futura de alguno de los parámetros clave del problema en función de los datos históricos que hubiera disponibles hasta la fecha. Dentro de los análisis de tipo descriptivo se pueden distinguir dos tipos de procesos o tareas, las de agrupamiento y las de asociación. Las primeras persiguen crear grupos de ítems en torno a uno de ellos, el representativo del grupo; mientras que las de asociación identifican patrones importantes dentro del conjunto global de datos. Asimismo, los análisis de tipo

predictivo también engloban dos tipos de tareas, las de clasificación y las de regresión. Un proceso de clasificación busca asignar una etiqueta (clase) a un determinado ítem en función de los valores de los parámetros que lo describen; por otra parte, en un proceso de regresión se busca hacer algo parecido, pero asignando al ítem no una etiqueta o valor de clase sino un valor numérico. Podría decirse entonces que un proceso de clasificación realiza predicciones cualitativas, mientras que uno de regresión las hace cuantitativas.

Tabla 1.1: Clasificación de métodos de Minería de Datos (fuente: Rabasa 2009)

			TIPOS DE ANÁLISIS			
			Descriptivo		**Predictivo**	
	TAREAS		**Agrupamiento**	**Asociación**	**Clasificación**	**Regresión**
MÉTODOS	**RR.NN.**		Back Propagation		Back Propagation	Back Propagation
	S. Regresión Lineal / No Lineal					Lineal / No Lineal
	Logística			Regresión Logística	Regresión Logística	
	Redes Bayesianas			Redes Bayesianas	Redes Bayesianas	
	Algoritmos Genéticos		Algoritmos Genéticos	Algoritmos Genéticos	Algoritmos Genéticos	Algoritmos Genéticos
	Clustering		Kmeans Kmedoids CLARA			
	Conteo de frecuencias		Apriori AprioriTid			
	Árboles de decisión	**Clasific.**	ID3 C4.5		ID3 C4.5	
		Regresión			CART M5	CART M5

Originalmente, para realizar cada tipo de análisis, descriptivo o predictivo, se ha utilizado un determinado método que en su momento fue ideado y diseñado para tal fin. Tendencias más actuales sostienen que la frontera que delimita ambos tipos de análisis no es todo lo estricta que en un principio parecía (p.e.: un agrupamiento puede verse como un tipo particular de clasificación); por lo que muchos métodos de análisis que en su momento se idearon con una finalidad predictiva, se ha visto que pueden dar también buenos resultados en procesos descriptivos. El caso contrario, aunque menos común, también se puede dar.

Así, por ejemplo, y en el ámbito que concierne a esta Tesis, los algoritmos generadores de árboles de decisión, del tipo ID3 (Quinlan 1979) o C4.5 (Quinlan 2006), que implementan métodos de clasificación, pueden ser utilizados con finalidad tanto descriptiva como predictiva. Es decir, se puede aplicar un método de generación de un árbol de decisión para crear un modelo que permita predecir a qué clase pertenece un determinado ítem, o bien, se puede aplicar ese mismo método para tratar de agrupar los ítems existentes en torno a la variable de clase que el método les asigne en cada caso.

1.2. Técnicas predictivas de Data Mining: Árboles de decisión y sistemas de reglas

Esta Tesis se centra en las Reglas de Clasificación (método de finalidad básicamente predictiva), pero antes de entrar en detalle, y con el único objeto de contextualizar este trabajo, es conveniente introducir algunas generalidades sobre las técnicas predictivas más utilizadas en el ámbito científico y empresarial.

Uno de los modelos predictivos más empleados en Minería de Datos son los árboles de decisión, que dependiendo de la naturaleza de la variable objetivo pueden ser: árboles de clasificación (variable objetivo de naturaleza nominal o discreta) y árboles de regresión (variable objetivo de naturaleza numérica o continua). Existen algunos árboles capaces de llevar a cabo tanto tareas de clasificación como de regresión, por ejemplo CART (Breiman et al. 1984) o M5 (Yang et al. 2002).

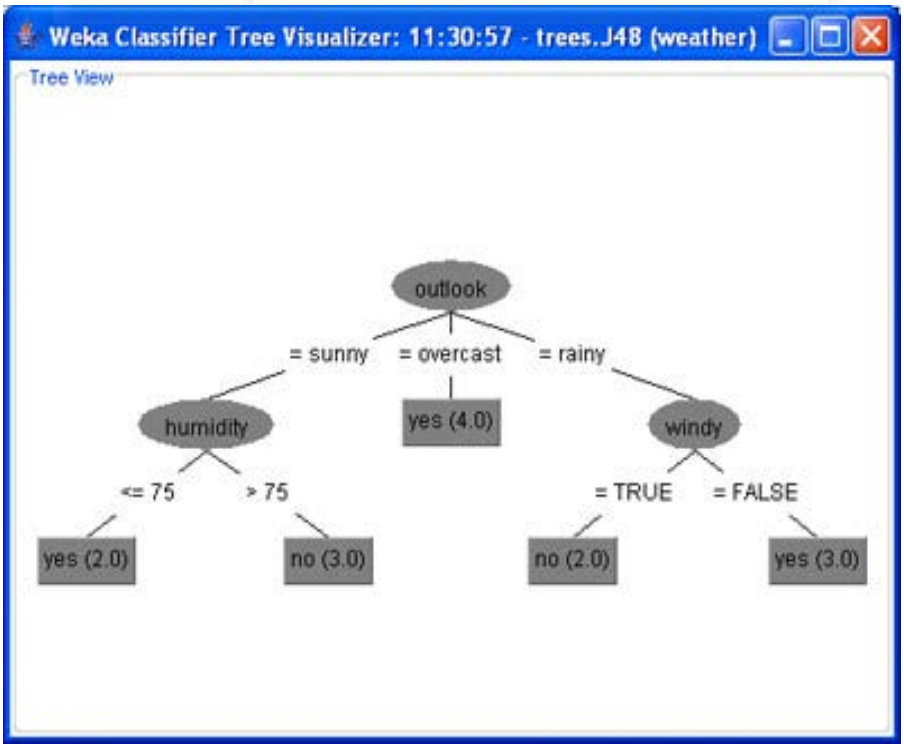


Figura. 1.1: Ejemplo de árbol de clasificación (WEKA)

La figura 1.1 muestra una ventana de la aplicación WEKA en la que se representa la salida gráfica correspondiente a un árbol de clasificación J48 (implementación Java del árbol C4.5), capaz de procesar información discreta y también numérica en sus atributos de entrada. Sobre el árbol, se puede interpretar cada rama, desde la raíz (nodo superior) hasta cada hoja (nodos terminales), como una regla de clasificación, donde los diferentes atributos se van instanciando a posibles valores. Cada nodo hoja viene acompañado del número total de ocurrencias que corresponde a tal regla (o de su porcentaje de aparición sobre el total de tuplas de entrada).

Existen múltiples implementaciones tanto para árboles de clasificación como para árboles de regresión. Tales implementaciones se diferencian, fundamentalmente, en los heurísticos de poda que emplean, los criterios de expansión que utilizan y formas de recorrido. Pero, en general, la mayor diferencia desde el punto de vista del usuario/analista radica en la naturaleza de las variables (en este caso de las ramas). Así, hay árboles como el ID3 (Quinlan 1979) que trabajan únicamente con variables nominales (tanto en los nodos internos como en las hojas), mientras que otros, como el C4.5 (Quinlan 2006) también manejan variables numéricas en sus nodos internos.

Centrándonos en problemas de clasificación (tal y como nos ocupa en esta Tesis), el recorrido en profundidad de cada rama de un árbol, desde el nodo raíz hasta una de sus hojas, produce una regla de clasificación. Además, existen métodos de generación de reglas, derivados del clásico conteo de frecuencias, en los que se van generando reglas que cumplen un determinado soporte de ocurrencia.

1.3. Selección automática de características

La selección automática de características (FS, '*Feature Selection*') es un conjunto de técnicas analíticas que descubren las combinaciones de atributos que tienen una mayor incidencia en el comportamiento de una determinada variable objetivo. La extracción de características resulta especialmente importante en problemas donde intervienen muchos atributos y las muestras de datos no son demasiado grandes, es decir, problemas "anchos" (Dernoncourt et al. 2014) y (Hall, P., Xue, J. 2014). La extracción de características relevantes puede ser llevada a cabo aplicando diferentes metodologías como el SVM (Support Vector Machine) y técnicas de Análisis de Componentes Principales (Martínez-Murcia et al. 2014), con un enfoque incremental donde los algoritmos empleados buscan ir mejorando una determinada métrica de significancia (Lu et al. 2014). Centrándonos en el ámbito empresarial, la extracción automática de características se ha relevado crítica en diferentes situaciones como en CRM´s ('Customer Relationship Management' o 'Gestión de Relaciones con Clientes') de compañías aéreas (Ismail and Husnayati 2013), en predicción de quiebra en sistemas financieros (Tsai 2009), o en predicción de rotura de stock (Tsai and Hsiao 2010).

1.4. Introducción a la Inteligencia de Negocio

El primer autor de que se tiene constancia que hiciera uso del término Business Intelligence (o 'B.I.') fue Hans Peter Luhn (Luhn 1958). En su trabajo habla de los 'sistemas automáticos' (entiéndase, 'sistemas informáticos' o 'sistemas de información') como herramientas para difundir información entre las diferentes secciones de cualquier organización y califica a los mismos como 'sistemas de inteligencia' que realizan estos tres tipos de tareas: auto-abstracción de documentos, auto-codificación de documentos y creación y actualización de perfiles de puntos de acción. Con ello hace referencia al proceso por el cual la información es capturada, codificada y distribuida allí donde es necesaria. Un sistema de estas características siempre debe ofrecer una especial flexibilidad para identificar la información conocida, buscar la que se necesita y difundirla de forma eficiente. El propio Luhn reconocía, hace más de medio siglo, algunos de los problemas que hoy día siguen siendo discutidos en el ámbito de la Inteligencia de Negocios como el "aumento del ritmo de generación y utilización de la información", "el crecimiento de las organizaciones y la mayor especialización como creadores de nuevas barreras para la difusión de información", "la necesidad de los órganos de responsabilidad de tomar cada vez más decisiones y en menos tiempo" o "la rápida distribución de la información en los lugares apropiados cuando se demande (Luhn ya usó el término 'on demand')"; en definitiva, la finalidad de estos sistemas debe ser, en sus propias palabras, "permitir al usuario acceder rápidamente a toda la información que necesite allí donde la necesite".

Más tarde, en 1989, Howard Dresner popularizó el término Business Intelligence (lo que erróneamente ha hecho que muchos le consideren como la primera persona en acuñarlo); este autor definió[1] la Inteligencia de Negocios del siguiente modo:

> "*Es el conjunto de conceptos y métodos para mejorar la toma de decisiones en los negocios utilizando sistemas de apoyo basados en hechos*"

En la definición anterior, se entiende por 'hecho' toda transacción susceptible de ser registrada (una compra, una venta, una orden de producción, una factura, etc...). En este punto entran en juego los Sistemas de Procesamiento de Transacciones, o TPS ('Transaction Processing Systems') y su posterior evolución a los sistemas de Procesamiento de Transacciones en Línea o OLTP ('On-Line Transaction Processing'). Estos son el primer sistema de información que se suele implantar en cualquier organización; están pensados precisamente para registrar cualquier transacción que ocurra en el negocio con el fin de tener una vista completa y actualizada del mismo. La finalidad principal de los sistemas OLTP es agilizar al máximo el registro de todas las

[1] Curiosamente, no se ha encontrado ningún artículo de Dresner donde desarrolle su idea, aunque sí muchos documentos donde se le cita como autor de esta definición (por ejemplo: Gibson et al 2004).

transacciones que ocurran en la organización, es decir agilizar las operaciones de alta, baja y modificación de registros en la base de datos. Para cumplir con esta finalidad estos sistemas deben contar con una base de datos debidamente normalizada para que no haya información redundante ni inconsistencias. Un sistema así diseñado es muy eficiente para la realización de actualizaciones sobre la base de datos, pero no lo es tanto a la hora de ejecutar consultas, especialmente si estas son algo sofisticadas y el número de registros a considerar es elevado. Esta carencia viene a ser cubierta por los sistemas de Procesamiento Analítico en Línea o OLAP (‘*On-Line Analytical Processing*’). Estos sistemas distribuyen los datos históricos de una organización en cubos de información (o cubos OLAP) que pueden ser explorados fácilmente de forma que un usuario/analista puede estudiar la evolución de una determinada variable (p.e. ‘volumen de ventas’), en función de diversos parámetros y diversos niveles de agregación, como pueden ser: en función del tiempo, por semanas, meses o temporadas; en función del producto, por tipos de productos, familias o gamas; en función de los lugares de venta, por punto de venta, localidades, regiones o países; etc. También sería posible hacer análisis más elaborados combinando varios parámetros e incluso obtener resultados comparativos (p.e. de distintos años o distintos productos).

Tradicionalmente, hablar de Inteligencia de Negocios a nivel práctico ha consistido en hablar de herramientas OLAP, y de cómo facilitar el uso de dichas herramientas a ejecutivos sin conocimientos técnicos sobre administración de bases de datos, así es como surgen los denominados Sistemas de Información para Ejecutivos o EIS (‘*Executive Information Systems*’). Estos sistemas proporcionan una vista más amigable que permite explotar, sin usar tecnicismos, una herramienta de análisis de datos más sofisticada. Típicamente, dicha herramienta ha consistido en una herramienta OLAP.

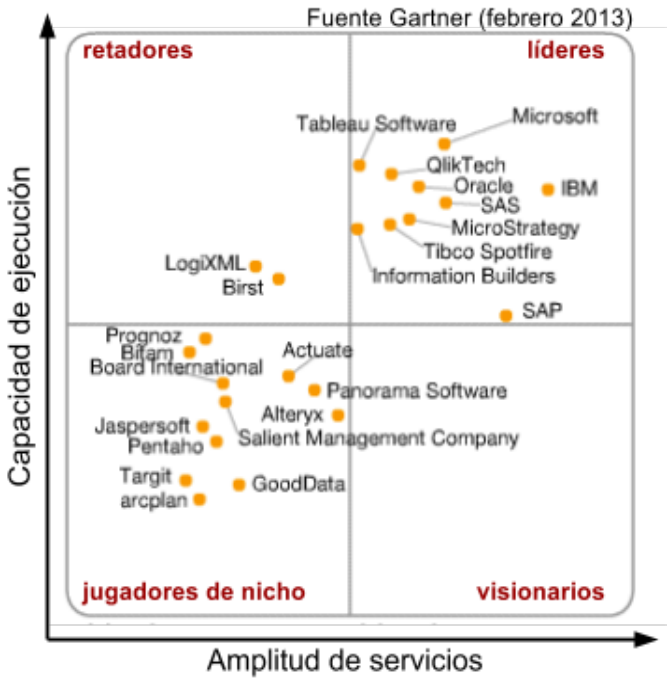


Figura 1.2: Clasificación de empresas desarrolladoras de software de B.I.

Las herramientas comerciales de Inteligencia de Negocio existentes ofrecen muy distintas posibilidades de análisis de datos, bien desde el enfoque OLTP, bien mediante OLAP. La figura 1.2 presenta un esquema de las empresas líderes en desarrollo de soluciones de B.I. El eje de las X representa el grado de completitud de su visión de la realidad del B.I., mientras que el eje de las Y representa la capacidad de ejecución de funcionalidades propias de esta disciplina.

Algunas de las herramientas de estas compañías incorporan técnicas de Minería de Datos que facilitan el análisis descriptivo de los mismos. Entre ellas, las más extendidas en nuestro ámbito empresarial más próximo son Pentaho y QlikView (de QlikTech), especialmente potentes en el desarrollo y mantenimiento de cuadros de mando integrales (figura 1.3).

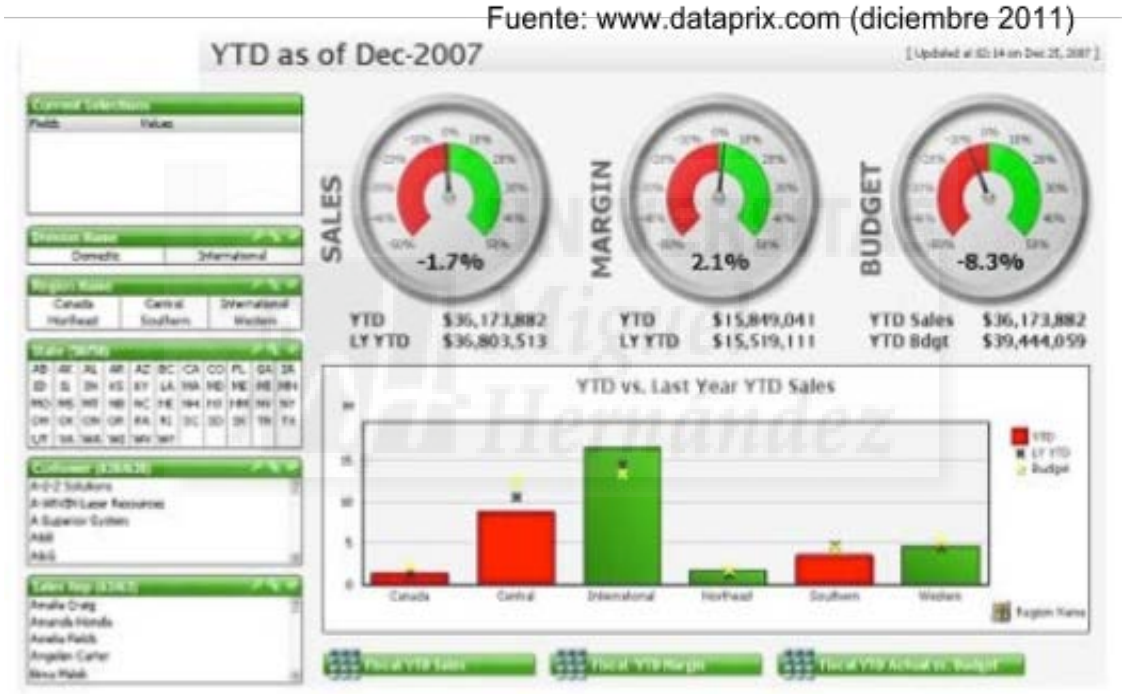


Figura 1.3: Ejemplo de Cuadro de Mando con QlikView

Sin embargo, desde el punto de vista puramente analítico, la principal carencia que las caracteriza es su limitada capacidad predictiva, así como la escasez de automatismos a la hora de seleccionar los atributos estadísticamente más relevantes, de cara al modelado de una variable objetivo. Dicha carencia es perfectamente subsanable con la incorporación de técnicas de Minería de Datos (‘*Automatic Feature Selection*’) que son perfectamente complementarias y aportan gran valor a las herramientas existentes de Inteligencia de Negocio desde el punto de vista de la automatización y optimización de los análisis. Con la incorporación de dichas técnicas, la selección de variables a considerar en cada escenario ya no dependerá exclusivamente de la experiencia y costumbres del analista al cargo de la herramienta.

Recientemente, ha aumentado considerablemente la cantidad de publicaciones científicas que ponen de manifiesto que el desarrollo de este tipo de aplicaciones y el desarrollo formal previo que las posibilita, es un área de investigación de muchísima proyección, desde las universidades de todo el mundo. Así pues, hay trabajos en que se pone el foco de atención en el desarrollo de una metodología capaz de analizar flujos continuos de datos desestructurados que provienen del exterior de la empresa y que, debidamente correlacionados, pueden aportar un gran valor para la organización (Castellanos et al. 2012). En definitiva, la repentina aparición de herramientas de B.I. en la última década, ha sido tan abrumadora, que algunos investigadores (Seng and Chiu 2011) se han planteado la necesidad de diseñar herramientas de *benchmarking* capaces de medir las capacidades de tales productos, así como sus requerimientos hardware y software que les permitan ser realmente operativos.

1.5. Técnicas de Data Mining aplicadas a la Inteligencia de Negocio

Son varias las disciplinas que han ofrecido algunas de sus técnicas más características para la resolución de problemas de diferente naturaleza dentro del B.I. Así por ejemplo, es frecuente encontrar la aplicación de técnicas estadísticas. Algunos autores (Awad et al. 2012) aplican Lógica Temporal Lineal para resolver problemas típicos del proceso general de negocio. En otros trabajos (Pan 2012) se apuesta por el uso de Análisis Multivariante y Regresión Logística en el ámbito de las finanzas, aplicando técnicas más próximas ya a la Inteligencia Artificial como las Redes Neuronales y el Clustering. Tanto una como la otra se han utilizado asiduamente en problemas de segmentación de clientes (Romdhane et al. 2010) y también para caracterizar escenarios en problemas de e-commerce (Chou et al. 2010) y (Qu and Liang 2009).

Pero es la Minería de Datos la disciplina científica que ofrece un abanico más amplio, tanto de técnicas como de tipología de problemas a resolver dentro del ámbito que nos ocupa. Empezando por una perspectiva más general, una gran variedad de trabajos analizan el uso de Minería al servicio del proceso general de negocio. Así pues, hay trabajos (Sharma et al. 2012) que presentan los resultados de una comprobación del proceso de integración de Minería de Datos dentro del proceso genérico de *Knowledge Discovery* vigente en cualquier modelo de negocio. Dentro de este mismo marco, (Seng and Chen 2010) también se ahonda en un enfoque analítico de cómo orientar el proceso de Minería de Datos dentro del contexto de negocio, y presenta un modelo para la selección de los métodos de Minería de Datos más adecuados en cada caso. En otros trabajos (Pugna et al. 2011) se resalta el salto cualitativo que favorece la Minería de Datos en el contexto de negocio, al permitir pasar de sistemas descriptivos (más o menos complejos) a modelos predictivos que aportan mucho más valor estratégico a los directivos.

Otros estudios contemplan la Minería de Datos (en general) para resolver problemas en facetas concretas de la empresa. Así, unos trabajos abordan la mejora en el manejo de las bases de datos de las compañías (Aggarwal et al. 2012). Otros trabajos se centran en explicar cómo la Minería supone una ventaja para la explotación de herramientas OLAP (Ali and Mouakket 2010). Otra aportación genérica muy interesante, en este caso orientada al e-commerce, es la ofrecida en (Liu et al 2011).

Una de las técnicas de Minería que más se ha empleado en el ámbito empresarial son las Reglas de Asociación, que descubren patrones de comportamiento entre los datos. Existen trabajos que aplican esta técnica también al proceso general de negocio de la compañía (Huang et al. 2011), (Kuosa 2011) y (Lim and Lee 2010), pero existen muchos otros estudios que tratan de descubrir patrones en dominios mucho más específicos. También se extraen patrones dentro de los llamados valores intangibles de la compañía (Shih et al. 2010) o dentro de extensas bases de datos de gestión (Lee et al. 2012). Otros autores se centran en la extracción de patrones en portales de comercio electrónico (Huang 2012) y (Chang and Chen 2012). Otro tipo de reglas, las de clasificación, normalmente obtenidas a partir de la construcción de árboles de decisión, se utilizan como una herramienta fundamental en el proceso general de negocio (Folino et al. 2011), (Li et al. 2010), (Lee 2010) y (Lim and Lee 2010). Más concretamente, las reglas de clasificación y árboles de decisión se han aplicado con éxito para predecir el comportamiento de valores intangibles de las compañías (Tsai et al. 2012), en las finanzas (Lin et al. 2011), en el pronóstico del comportamiento de clientes desde diferentes perspectivas (Romdhane et al. 2010), (Liou et al. 2011) y (Peng and Xu 2011); y por supuesto, en el pronóstico de tendencias y preferencias en portales de comercio electrónico (Cheung and Li 2012) y (Jun 2012).

Muy frecuentemente, la gran cantidad de variables de entrada en los problemas de Inteligencia de Negocio, hace necesario recurrir a la extracción de características más relevantes en cada problema. En este sentido, algunos autores incorporan este tipo de técnicas sobre los valores intangibles del negocio (Tsai et al. 2012), otros lo hacen en el terreno de las finanzas (Lin et al. 2011), sobre las variables concernientes a los clientes (Goedertier et al. 2011) y (Romdhane et al. 2010), o para mejorar la explotación de las herramientas OLAP (Zhang et al. 2011).

Una de las técnicas de Minería de Datos más prometedoras, en lo que a contextos de optimización se refiere, son los Algoritmos Genéticos. El empleo de esta técnica permite, entre otras cosas, reducir también el número de variables a tener en cuenta en un problema, filtrar las reglas y patrones generados para seleccionar únicamente los más importantes en cada caso o determinar los grupos más significativos con objeto de realizar agrupamientos óptimos de cualquier ente del problema de negocio. Así, algunos autores (Goedertier et al. 2011) aplican Algoritmos Genéticos en problemas de comercio electrónico para descubrir eventos importantes en el log del servidor de comercio, como por ejemplo trazas incompletas, comportamientos duplicados de los usuarios, etc.

Como conclusión de la investigación realizada en este estado del arte se podrían resaltar los siguientes aspectos:

- El uso de técnicas de Minería de Datos está aportando un gran valor añadido a las técnicas clásicas de explotación de la información en problemas de Business Intelligence, de muy diversa naturaleza.

- Debido al enorme auge del comercio electrónico y los grandes volúmenes de datos que es necesario explotar en esos escenarios, uno de los problemas más estudiados en los últimos años es la búsqueda de reglas y patrones en dichos portales; así como sobre las bases de datos de históricos de pedidos, sea cual sea el medio.

- Dado el gran número de variables (o atributos) que intervienen en la mayoría de problemas de Inteligencia de Negocio, una importante línea de investigación, en la actualidad, es el desarrollo de diferentes técnicas de extracción automática de características para seleccionar los atributos más importantes en cada caso, y a partir de ellos obtener sistemas de reglas más precisos que modelen el comportamiento de alguna de las variables intervinientes.

Por último, la tabla 1.2 contiene de forma esquemática un cuadro/resumen de los trabajos más significativos comentados en el presente epígrafe, clasificándolos según la técnica empleada y el ámbito de aplicación.

Tabla 1.2: Cuadro resumen – aplicación de técnicas de análisis de datos al ámbito empresarial

		ÁMBITO DEL NEGOCIO						
	Método / Procedimiento	**Proceso general de necio**	**Valores intangibles**	**BD de Gestión Empresarial**	**Finanzas**	**Clientes**	**OLAP**	**E-commerce**
Plataformas y Aplicaciones		Castellanos et al. 2012 Seng and Chiu 2011						
Estadística Clásica	**Log. Temporal Lineal**	Awad et al. 2012						
	A. Multivariante y Reg. Logística				Pan 2012			
Inteligencia Artificial	**Redes Neuronales**				Pan 2012	Romdhane et al. 2010		Chou et al. 2010
	Clustering				Pan 2012	Romdhane et al. 2010		Qu and Liang 2009
Minería de Datos	**Métodos Generales (DM y KDD)**	Sharma et al. 2012 Seng and Chen 2010 Pugna et al. 2011		Aggarwal et al. 2012			Ali and Mouakk et 2010	Liu et al 2011
	Patr./Reglas de Asociación	Huang et al. 2011 Kuosa 2011 Lim and Lee 2010	Shih et al. 2010	Lee et al. 2012				Huang 2012 Chang and Chen 2012
	Árb./Reglas de Clasificación	Folino et al. 2011 Li et al. 2010 Lee 2010 Lim and Lee 2010	Tsai et al. 2012		Lin et al. 2011	Romdhane et al. 2010 Liou et al. 2011 Peng and Xu 2011		Cheung and Li 2012 Jun 2012
	Selección de Características		Tsai et al. 2012		Lin et al. 2011	Goedertier et al. 2011 Romdhane et al. 2010	Zhang et al. 2011	
	Algoritmos Genéticos							Goedertier et al. 2011

UNIVERSITAS
Miguel
Hernández

Capítulo 2
Problema a resolver: Hipótesis de partida y objetivos

2.1. Justificación e hipótesis de partida

La dificultad de extraer automáticamente las variable más incidentes en cualquier proceso de Inteligencia de Negocio (justificado anteriormente en el epígrafe 1.3) desvela que el desarrollo de técnicas especialmente ágiles (bajo consumo de tiempo y memoria) para la extracción automática de características es, cada vez más, una necesidad para el descubrimiento y manejo de información en este campo. Este tipo de técnicas permitiría extraer reglas de comportamiento de los datos y ofrecerlas en poco tiempo, para ser estudiadas por el usuario interesado. Deben tenerse presentes las características especiales de dichos usuarios, a los que están dirigidas las aplicaciones de Inteligencia de Negocio. Normalmente, se trata de ejecutivos, gerentes y/o directores de empresas de un cierto tamaño, en general, personas que son responsables de la toma de decisiones estratégicas en sus respectivas organizaciones y que necesitan disponer de la información adecuada de forma rápida. Además, el nivel de penetración de las tecnologías de la información en este grupo de usuarios es cada vez mayor, por lo que es frecuente que dispongan de algún tipo de dispositivo móvil y que lo usen habitualmente como herramienta de trabajo. Algunas de las limitaciones más importantes que tienen estos dispositivos son la cantidad de memoria de que disponen, su reducida capacidad de cálculo (comparada con la de un PC por ejemplo), su autonomía (al funcionar con baterías que necesitan ser recargadas periódicamente) y su

conectividad (ya que no siempre se dispone de una conexión de calidad suficiente). Un sistema de información diseñado para satisfacer las necesidades de este colectivo podría constar de un servidor de cálculo encargado de realizar las operaciones computacionalmente más costosas, y posteriormente podría mandar la salida del procesamiento efectuado a un dispositivo móvil para que el usuario final pueda ver los resultados del análisis realizado. Con unos algoritmos eficientes que proporcionen modelos de tamaño reducido, la información (también reducida) proporcionada por dichos algoritmos, puede mandarse telemáticamente en menos tiempo a un dispositivo móvil, y esté, con una aplicación adecuada, podrá renderizar los resultados. Obviamente, cualquier innovación que permita satisfacer mejor las necesidades dentro del ámbito de la Inteligencia de Negocios es automáticamente aplicable a otros campos o tipos de problema, es decir, una solución que hace viable, de manera ágil, la explotación de datos y su posterior estudio en dispositivos con capacidades reducidas, también mejorará los resultados en otro tipo de entornos.

En esta Tesis se presenta el algoritmo de Minería de Datos denominado CREA (‘*Classification Rules Extraction Algorithm*’) que genera y reduce sistemas de reglas de clasificación y consigue predicciones tan precisas como los métodos existentes hasta la fecha, pero con un coste temporal y de consumo de memoria sustancialmente menor. Si bien el planteamiento de partida se circunscribe al ámbito de la Inteligencia de Negocio, dado que es un claro exponente de ambas restricciones, la aplicabilidad del método propuesto, tal y como se ha comentado, no queda reducida, exclusivamente, a dicho ámbito. No obstante, el método propuesto, como cualquier método de Minería de Datos y con independencia del ámbito final de aplicación, será absolutamente dependiente de las muestras de datos que recibe a su entrada: no sólo de un tamaño mínimo exigible para poder inferir comportamientos, sino también de la “calidad” de los mismos.

2.1.1. Inducción de árboles de decisión

Como ya se indicó en el epígrafe 1.2, el método de Minería de Datos más conocido para la generación de reglas de clasificación, donde tanto los atributos que forman parte del antecedente como la variable consecuente son de tipo nominal (no-numérico), es el árbol de clasificación ID3 de Ross Quinlan (Quinlan 1979). En un trabajo posterior (Quinlan 1986), el autor identifica su algoritmo como perteneciente a la familia de sistemas de aprendizaje TDIDT (‘*Top-Down Induction Decision Trees*’, algoritmo 2.1). Estos métodos reciben como entrada un conjunto de ejemplos ‘E’ (ítems, ver figura 2.1) caracterizados por una serie de atributos que toman unos determinados valores y que tienen, cada uno de ellos, asignada una variable de clase (consecuente). El objetivo de estos métodos es inducir un árbol de decisión a partir del conjunto de datos de entrada aplicando los pasos del algoritmo TDIDT. En dicho algoritmo se va construyendo un árbol de forma recursiva en el que cada nodo del árbol es etiquetado con un atributo ‘A’ del conjunto de ejemplos ‘E’ y cada rama con uno de los posibles valores ‘V_i’ del atributo del nodo padre. Se define como caso base de la recursión la condición de si

todos los ejemplos de un determinado nodo pertenecen a la misma clase 'C', si es el caso, el nodo generado se etiqueta como nodo final u hoja. Si no es el caso, se selecciona el '*mejor atributo*' 'A' del conjunto de ejemplos y entonces 'E' se divide en tantos subconjuntos 'E_i' como posibles valores 'V_i' pueda tomar el atributo 'A' y se repite el proceso recursivamente para cada uno de los subconjuntos 'E_i'.

Algoritmo 2.1: TDIDT (E: Conjunto de ejemplos)

```
1.  N ← nuevo NODO
2.  Si todos los ejemplos de E pertenecen a la misma clase C Entonces
3.          N.clase = C
4.          N.esHoja = TRUE
5.  Si no
6.          Seleccionar el mejor atributo A del conjunto E
7.          Nodo.atributo = A
8.          Para cada valor Vi de A
9.            Ei ← Subconjunto de E donde A = Vi
10.           Nodo.valorRamai = Vi
11.           Nodo.ramai = TDIDT ( Ei )
12.         Fin Para
13. Fin Si
14. Devolver N
```

	N°	Vista cielo	Temperatura	Humedad	Viento	CLASE
	1	Soleado	Calor	Alta	No	No Jugar
E_i (i=2)	2	Soleado	Calor	Alta	Sí	No Jugar
	3	Nublado	Calor	Alta	No	Jugar
	4	Lluvia	Intermedia	Alta	No	Jugar
	5	Lluvia	Frío	Normal	No	Jugar
	6	Lluvia	Frío	Normal	Sí	No Jugar
	7	Nublado	Frío	Normal	Sí	Jugar
	8	Soleado	Intermedia	Alta	No	No Jugar
	9	Soleado	Frío	Normal	No	Jugar
	10	Lluvia	Intermedia	Normal	No	Jugar
	11	Soleado	Intermedia	Normal	Sí	Jugar
	12	Nublado	Intermedia	Alta	Sí	Jugar
	13	Nublado	Calor	Normal	No	Jugar
	14	Lluvia	Intermedia	Alta	Sí	No Jugar

E: filas 1–14 · Atributos: Vista cielo, Temperatura, Humedad, Viento · $\{V_i\}$: Frío, Intermedia, Calor · A: Temperatura · C: CLASE

Figura 2.1: Conjunto de ejemplos ('E') para generar árbol de decisión (clásico)

La clave del algoritmo 2.1 se encuentra en el paso 6 '*Seleccionar el mejor atributo A*'. En este punto hay que plantear cuál es el criterio por el que se identifica un atributo como '*mejor*' que el resto. Se considerará mejor aquella elección de atributos que acaben generando un árbol que cumpla las siguientes dos condiciones: que clasifique correctamente todos los ejemplos del conjunto de datos 'E', y que el árbol sea lo más reducido posible, entendiendo por reducido, que tenga el menor número de nodos y ramas posible. Inducir árboles que cumplan la primera condición es una tarea relativamente sencilla. En la figura 2.2, se muestra un árbol demasiado complejo,

ejemplo de la situación que se acaba de describir, inducido a partir de los ejemplos del conjunto ‘E’ (figura 2.1).

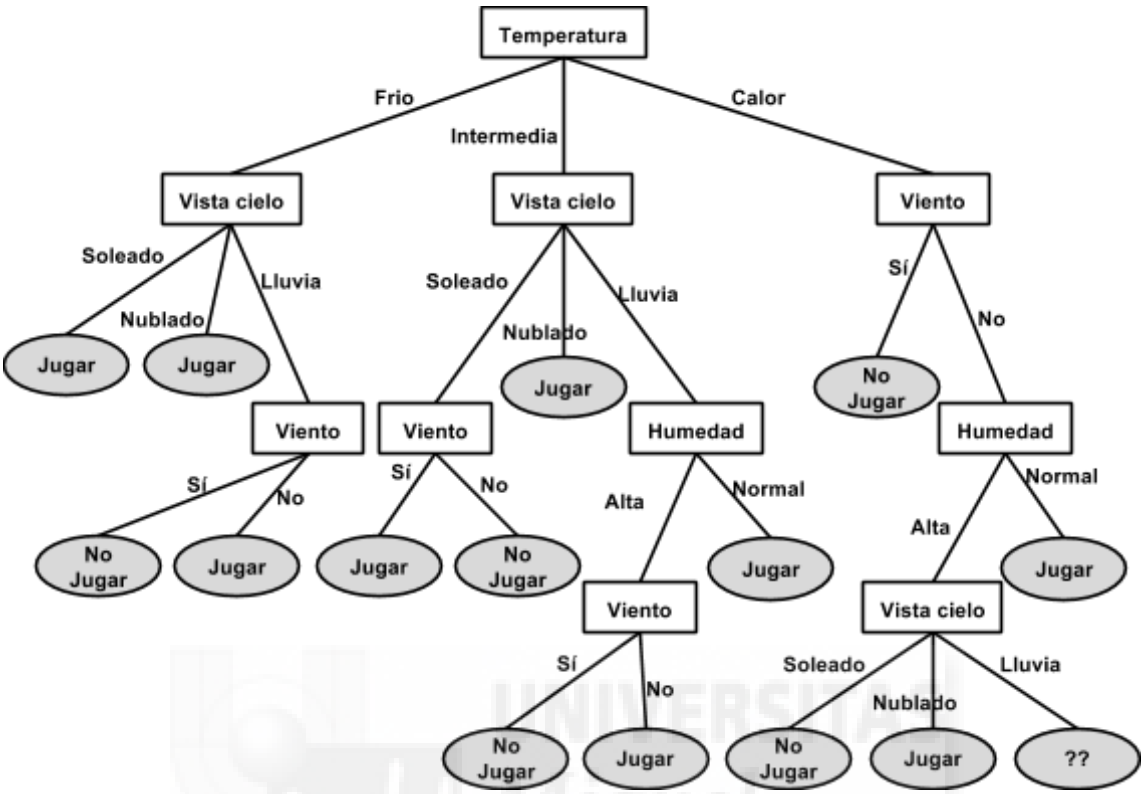


Figura 2.2: Árbol de decisión complejo (muy extenso)

Una solución óptima a este problema pasaría por generar todos los árboles que cumplan la primera de las condiciones (fuerza bruta) y de entre ellos seleccionar como resultado final el más reducido, pero esta alternativa tendría un coste computacional excesivamente alto, inviable para proporcionar una respuesta en un tiempo razonable cuando se analizan grandes volúmenes de datos.

El algoritmo TDIDT sigue un esquema voraz (greedy) según el cual, se debe disponer de algún mecanismo para decidir de forma automática cuál debe ser el atributo por el que seguir expandiendo el árbol en cada momento. Quinlan propone, para su método ID3, utilizar como criterio el concepto de ‘*ganancia de la información*’, basado en la teoría de la información de Claude E. Shannon (Shannon, 1948). Si los ítems de un conjunto de ejemplos ‘E’ están clasificados como ‘$Clase_1$’, ‘$Clase_2$’, ..., ‘$Clase_n$’ de tal forma que las probabilidades de que se encuentren elementos de cada una de las clases son ‘p_1’, ‘p_2’, ..., ‘p_n’ respectivamente, se define el concepto de entropía de información ‘I(E)’ como:

$$I(E) = -\sum_{i=1}^{n} p_i \log_2(p_i) \qquad \text{(Expresión 2.1)}$$

Para un atributo ‘A’ de dicho conjunto que puede tomar los valores ‘A_1’, ‘A_2’, ..., ‘A_v’, se definen las particiones ‘E_1’, ‘E_2’, ..., ‘E_v’ del conjunto ‘E’ tal que cada partición ‘E_i’

contiene los ejemplos de 'E' con valor 'A_i' para el citado atributo 'A'. Se define entonces la ganancia de información del atributo 'A', 'G(A)' como:

$$G(A) = I(E) - \sum_{i=1}^{v} p(E_i)\, I(E_i) \qquad \text{(Expresión 2.2)}$$

Donde '$p(E_i)$' es la probabilidad de encontrar un ítem del conjunto 'E_i' dentro de 'E'. Para seleccionar el '*mejor*' atributo por el que expandir cada nodo del árbol de decisión (paso 6 del algoritmo 2.1) el método ID3 elige aquel que maximiza la ganancia de información. Dado que para un mismo nodo del árbol la entropía de información ('I(E)') permanece constante, maximizar la ganancia de información equivale a minimizar el siguiente sumatorio.

$$\sum_{i=1}^{v} p(E_i)\, I(E_i) \qquad \text{(Expresión 2.3)}$$

Para ilustrar el funcionamiento del algoritmo ID3, véase el siguiente desarrollo aplicado al conjunto de datos de ejemplo 'E' de la figura 2.1:

Probabilidad de 'Jugar' ➔ p1 = 9/14
Probabilidad de 'No Jugar' ➔ p2 = 5/14

Entropía de información ➔ $I(E) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14) = 0.940$

$G(\text{Temperatura}) = I(E) - p(E_{Calor})\, I(E_{Calor}) - p(E_{Interm})\, I(E_{Interm}) - p(E_{Frio})\, I(E_{Frio})$

$I(E_{Calor}) = -2/4 \log_2(2/4) - 2/4 \log_2(2/4) = 1.0$
$I(E_{Interm}) = -4/6 \log_2(4/6) - 2/6 \log_2(2/6) = 0.918$
$I(E_{Frio}) = -3/4 \log_2(3/4) - 1/4 \log_2(1/4) = 0.811$

$G(\text{Temperatura}) = 0.94 - 4/14 \times 1 - 6/14 \times 0.918 - 4/14 \times 0.811 = 0.029$

Repitiendo los cálculos para el resto de atributos se tiene que:

G(Vista cielo) = 0.246
G(Humedad) = 0.151
G(Viento) = 0.048

Se observa que el atributo 'Vista cielo' es el que proporciona la mayor ganancia de información, por tanto será considerado como el '*mejor*'. Es decir, será el primer atributo por el que se empezará a expandir el árbol de decisión. A continuación se generarán tres conjuntos de datos, '$E_{Soleado}$', '$E_{Nublado}$' y 'E_{Lluvia}' y se repite el proceso de forma recursiva como indica el algoritmo TDIDT. Finalmente se obtiene un árbol de decisión como el de la figura 2.3. Este árbol, al igual que el de la figura 2.2, clasifica correctamente todos los ejemplos del conjunto de datos 'E' (figura 2.1), pero además, como se puede observar, está bastante más simplificado que el anterior, por lo que se le

considera una mejor solución. No se puede asegurar que sea la mejor de entre todas las posibles soluciones, pero con el algoritmo ID3 se ha proporcionado un heurístico que alcanza, en un tiempo razonable, una solución aceptablemente buena.

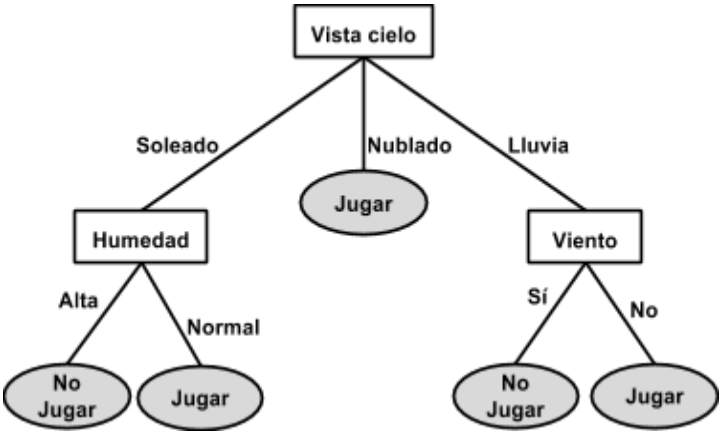


Figura 2.3: Árbol de decisión simplificado

El árbol de la figura 2.3 puede ser convertido fácilmente en un modelo de reglas de clasificación, en el que cada uno de los posibles caminos desde el nodo raíz hasta cada nodo hoja es una regla del tipo *SI 'antecedente' ENTONCES 'consecuente'*. En otras palabras, el árbol de la figura 2.3 sería equivalente al sistema de reglas de la tabla 2.1. Como se observa en dicha tabla, el modelo construido consta de 5 reglas, una por cada nodo hoja del árbol. De haber construido un sistema de reglas con el árbol más extenso de la figura 2.2 habrían resultado un total de 15 reglas (tantas como nodos hoja), dando lugar a un modelo más complejo y costoso de manejar.

Tabla 2.1: Sistema de reglas

ANTECEDENTE				CONSECUENTE
'Vista Cielo' = 'Soleado'	Y	'Humedad' = 'Alta'	➔	NO JUGAR
'Vista Cielo' = 'Soleado'	Y	'Humedad' = 'Normal'	➔	JUGAR
'Vista Cielo' = 'Nublado'			➔	JUGAR
'Vista Cielo' = 'Lluvia'	Y	'Viento' = 'Sí'	➔	NO JUGAR
'Vista Cielo' = 'Lluvia'	Y	'Viento' = 'No'	➔	JUGAR

Para determinar la complejidad temporal del algoritmo ID3 hay que tener en cuenta las siguientes consideraciones: en cada nodo interno del árbol (nodo no-hoja) se debe determinar la ganancia de cada uno de los atributos no testado anteriormente; si llamamos 'N' al número de ítems del conjunto 'E' y siendo 'C' el número de atributos (columnas), la ganancia en sí misma dependerá de la cantidad de ejemplos 'N' y del número de atributos 'C'. En (Quinlan 1986), la complejidad temporal asintótica del proceso en cada nodo viene dada por la expresión:

$$\Theta(N \times C) \qquad \text{(Expresión 2.4)}$$

Esta tarea se debe repetir en todos los nodos no-hoja del árbol, por lo que el factor de complejidad final del algoritmo quedará afectado por el número de estos. Se hace ver

que en el trabajo de Quinlan, a efectos de cálculo de la complejidad computacional del algoritmo, no se concreta nada más. No se considera cómo afecta el hecho de que las variables que definen los ítems del conjunto de ejemplos pueden tomar más o menos valores, influencia que queda patente en las expresiones 2.2 y 2.3. Tampoco dice nada acerca de cómo implementar, y que coste computacional y de memoria tendría, la segmentación del conjunto 'E' en los subconjuntos 'E_i' (paso 10 del algoritmo 2.1).

No obstante, el algoritmo ID3 se ha convertido en un estándar de facto en el ámbito de la generación de árboles de decisión o en la creación de modelos basados en reglas de clasificación. Han sido muchos los trabajos en los que se han introducido modificaciones en este algoritmo para dar lugar a otros con ciertas mejoras o mejor adaptados a determinados ámbitos. Así, por ejemplo, tenemos el algoritmo RID3 (Pal et al. 1997) que utiliza métodos genéticos para la generación de sucesivos árboles con umbral de rendimiento creciente. El ID3(+) (Xu et al. 2006) resuelve, entre otras cosas, la incapacidad de ID3 de manejar atributos con valores desconocidos o nulos. Otros estudios, proponen mejoras al método de poda original cuando los datos son deficientes o inciertos (Zhang et al. 2005), o en circunstancias de ramas con escasa fiabilidad (Zhang et al. 2008). Todos estos métodos para inducir árboles de decisión y sistemas de reglas de clasificación han resultado ser muy útiles, proporcionando magníficos resultados predictivos en ámbitos muy diversos, como por ejemplo en medicina pediátrica para extracción de señales predictivas de un catálogo de síndromes (Braaten 1996); en la detección de ataques web disminuyendo el ratio de falsos positivos (Folino et al. 2011) que se obtenían con métodos anteriores; en sistemas de alerta temprana '*anti-dumping*' (Zhao and Chang 2006) y muy a menudo, en problemas complejos de clasificación en el ámbito del marketing en sectores muy diversos (Zhang 2009), (Zhang 2008-BIS) y (Ke et al. 2007).

2.1.2. Reducción de conjuntos de reglas de clasificación

Los métodos generadores de sistemas de reglas de clasificación suelen producir conjuntos de reglas muy extensos. Cuando dichos conjuntos son empleados para construir un sistema experto o ES ('*Expert System*'), puede ser válido utilizar todas las reglas disponibles para implementar dicho sistema. Por el contrario, si el objetivo de un sistema de reglas es la construcción de un sistema de apoyo a la toma de decisiones o DSS ('*Decision Support System*'), se hace necesario reducir el número de reglas disponibles para que el usuario de ese DSS pueda trabajar con ellas. Adicionalmente, si se proporciona un mecanismo para ordenar las reglas, de forma que el usuario pueda saber cuáles de ellas son más confiables, en ese caso se estaría dando a dicho DSS un valor añadido que facilitaría aún más el trabajo del analista. Existen diferentes procedimientos en base a los cuales se pueden reducir los conjuntos de reglas generados por árboles de la familia de ID3. Algunos autores definen nuevas fórmulas para medir la Entropía de los atributos (Huang and Lin 1996); otros recalculan la importancia de los atributos que deben intervenir en la generación del árbol (Luo et al. 2010). Algunos

estudios, incluso, emplean técnicas de Lógica Difusa para reducir la sensibilidad original ante pequeños cambios en los valores de los atributos, y así obtener árboles de decisión más pequeños (Abu-halaweh and Harrison 2009). Para la realización de la presente Tesis se ha escogido como método de reducción y ordenación de reglas de clasificación el algoritmo RBS (Rabasa 2009). La figura 2.4 muestra como sería el proceso general de generación de un conjunto de reglas reducido empleando este método.

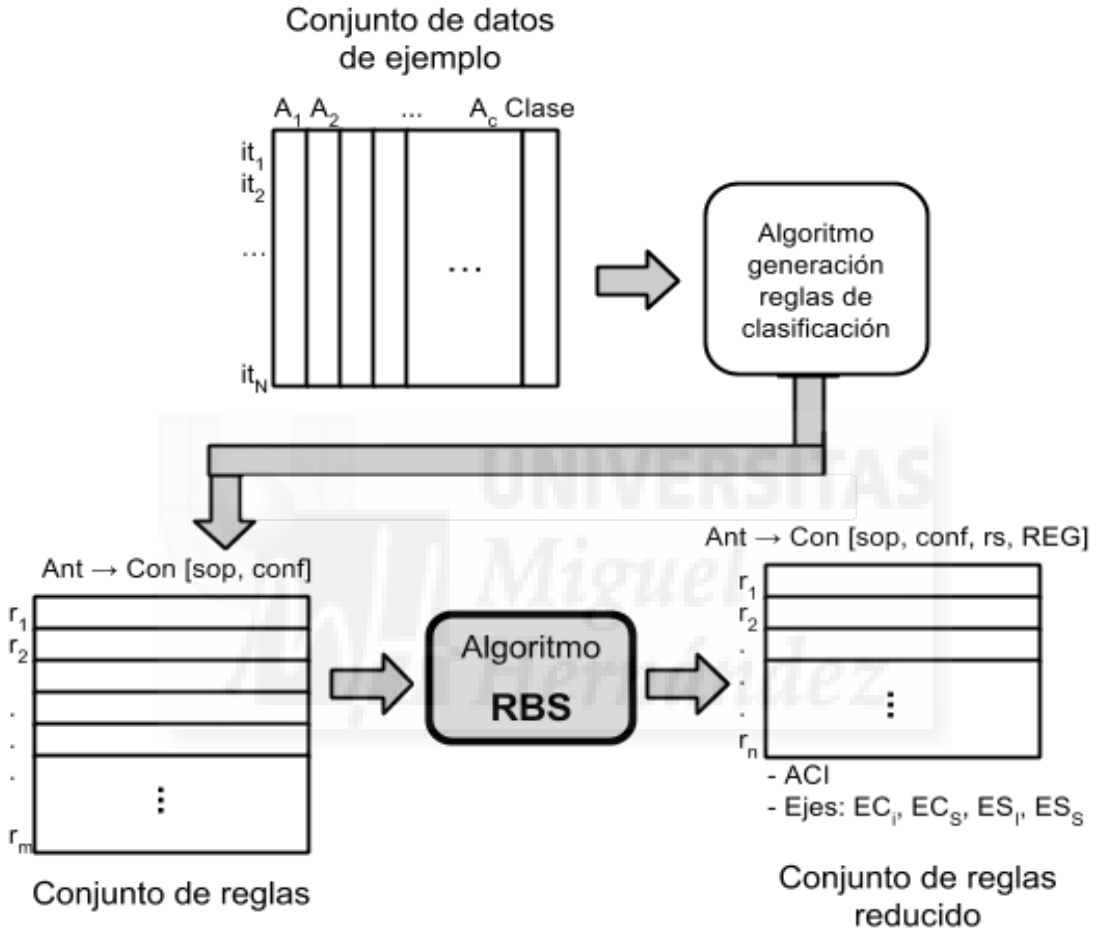


Figura 2.4: Proceso general de generación de conjuntos de reglas reducidos

Se parte de un conjunto de datos con 'N' ítems o ejemplos, caracterizados por una serie de 'C' atributos, donde cada ítem pertenece a una determinada 'Clase'. A partir de dicho conjunto de datos se genera un conjunto de reglas de clasificación, cada una de las cuales debe estar caracterizada por su soporte ('*sop*') y su confianza ('*conf*'), entendiendo por soporte de una regla la probabilidad de que se dé el antecedente de la misma en el conjunto de datos original, y por confianza la probabilidad de que se dé el consecuente dentro de los ítems del conjunto de datos original con el mismo antecedente. Por ejemplo, para la primera regla (en adelante 'r_1') de la tabla 2.1.

r_1: Vista cielo = 'Soleado' Y Humedad = 'Alta' ➔ NO JUGAR

Si se realiza el correspondiente conteo de frecuencias con ayuda de la figura 2.1, que contiene los datos del conjunto 'E' de entrenamiento que se ha utilizado para generar dicha regla, se tiene que el antecedente de 'r_1' (Vista cielo = 'Soleado' Y Humedad = 'Alta') se repite 3 veces en todo el conjunto 'E'; y en todos los ejemplos que cumplen el antecedente de 'r_1', el consecuente es siempre (las 3 veces) 'NO JUGAR', por tanto los valores de soporte y confianza para esta regla serán:

$$sop(r_1) = 3/14 = 0.214$$
$$conf(r_1) = 3/3 = 1.0$$

Así pues, la regla 'r1' quedará caracterizada tal que:

r_1: Vista cielo = 'Soleado' Y Humedad = 'Alta' ➔ NO JUGAR **[sop=0.214 , conf=1.0]**

Un conjunto de reglas 'R' así construido es la entrada necesaria para poder aplicar el algoritmo RBS. Nótese que al asignar un par de valores numéricos, soporte y confianza, a cada regla, estas se pueden representar en un plano como el de la figura 2.5 utilizando dichos valores de soporte y confianza como coordenadas.

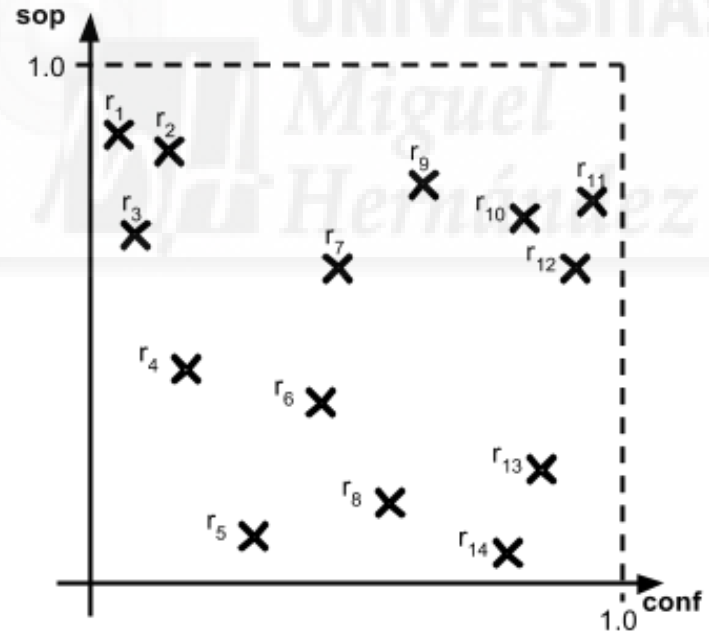


Figura 2.5: Representación en el plano de un conjunto de reglas de clasificación

El algoritmo RBS (ver algoritmo 2.2) empieza calculando, sobre dicho plano, los ejes 'Ec' y 'Es' ('*Eje Confianza*' y '*Eje Soporte*' respectivamente) tal y como se indica en la expresión 2.5.

$$Ec = 1 / |Clase|$$
$$Es = 1 / (|A_1| |A_2| \dots |A_C|) \qquad \text{(Expresión 2.5)}$$

Estos ejes dividen el plano conf|sop en dos áreas, izquierda y derecha, tal que ‘conf ≤ Ec’ y ‘conf > Ec’ respectivamente; y otras dos áreas, inferior y superior, tal que ‘sop ≤ Es’ y ‘sop>Es’ respectivamente. Identificadas estas áreas se definen los ejes ‘Ec_i’, ‘Ec_s’, ‘Es_i’ y ‘Es_s’ como:

$$\begin{aligned} Ec_i &= \text{Promedio}(\ \{\ conf(\ r_i\ )\ \}\ )\ \forall\ r_i : conf(\ r_i\ ) \le Ec \\ Ec_s &= \text{Promedio}(\ \{\ conf(\ r_i\ )\ \}\ )\ \forall\ r_i : conf(\ r_i\ ) > Ec \\ Es_i &= \text{Promedio}(\ \{\ sop(\ r_i\ )\ \}\ )\ \forall\ r_i : sop(\ r_i\ ) \le Es \\ Es_s &= \text{Promedio}(\ \{\ sop(\ r_i\ )\ \}\ )\ \forall\ r_i : sop(\ r_i\ ) > Es \end{aligned} \qquad \text{(Expresión 2.6)}$$

Con esos ejes, se definen las regiones ‘REG-0’, ‘REG-1’, ‘REG-2’ y ‘REG-3’ como:

$$\begin{aligned} \text{REG-1} &\Leftrightarrow sop \ge Es_s \text{ Y } conf \le Ec_i \\ \text{REG-2} &\Leftrightarrow sop \ge Es_s \text{ Y } conf \ge Ec_i \\ \text{REG-3} &\Leftrightarrow sop \le Es_i \\ \text{REG-0} &\text{ en otro caso} \end{aligned} \qquad \text{(Expresión 2.7)}$$

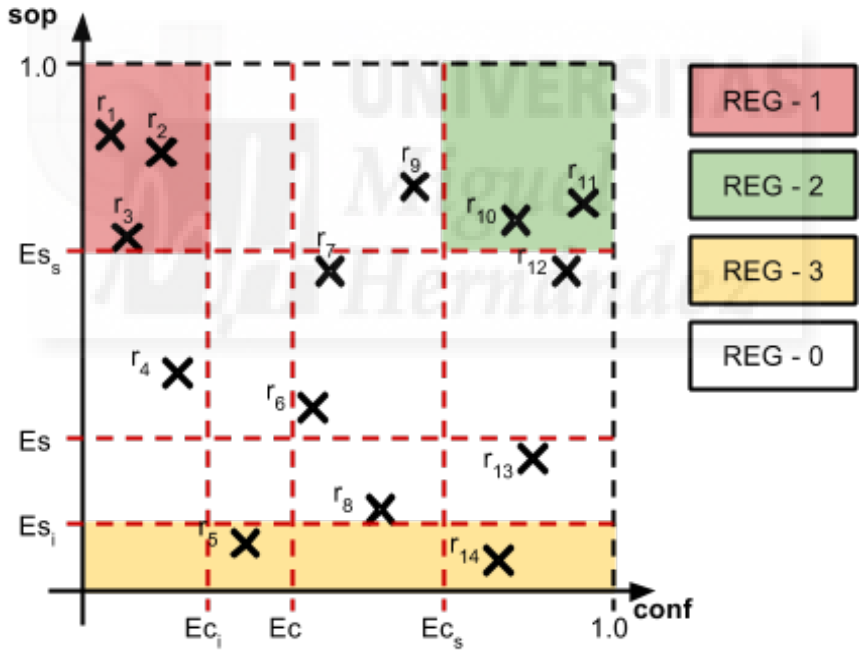


Figura 2.6: Representación gráfica de reglas de clasificación, ejes y regiones

Una vez que se han identificado las regiones, se procede a realizar un desplazamiento de los ejes ‘Ec_i’ y ‘Ec_s’, ajustándolos hacia la izquierda y hacia la derecha, respectivamente, para hacerlos coincidir con la regla ‘r_i’ más cercana que los aleje del eje de partida ‘Ec’, de esta forma se consigue una reducción de las áreas de las regiones ‘REG-1’ y ‘REG-2’. De forma análoga, los ejes originales ‘Es_i’ y ‘Es_s’ se ajustan hacia abajo y hacia arriba, respectivamente, para hacerlos coincidir con la regla ‘r_i’ más cercana que los aleje del eje de partida ‘Es’. El reajuste del eje ‘Es_i’ producirá una reducción del área ‘REG-3’, mientras que el desplazamiento del eje ‘Es_s’ vuelve a reducir las áreas de ‘REG-1’ y ‘REG-2’ (ver figura 2.7).

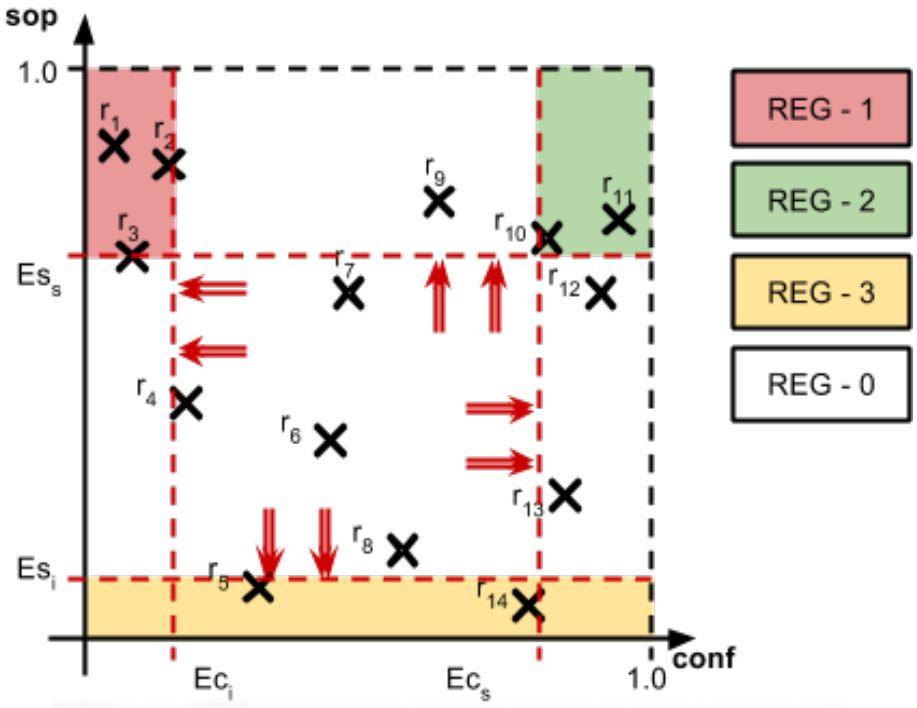


Figura 2.7: Ajuste de ejes y y reducción de las regiones 'REG-1', 'REG-2' y 'REG-3'

La finalidad de estos reajustes en los ejes que delimitan las regiones es reducir el área de las regiones 'REG-1', 'REG-2' y 'REG-3' conservando en todo momento el número de reglas que contenían inicialmente. Según las figuras 2.6 y 2.7, los mencionados ejes quedarían del siguiente modo:

$$\begin{aligned} Ec_i &= conf(\, r_2 \,) \\ Ec_s &= conf(\, r_{10} \,) \\ Es_i &= sop(\, r_5 \,) \\ Es_s &= sop(\, r_3 \,) \end{aligned} \qquad \text{(Expresión 2.8)}$$

A continuación, cada una de las reglas se etiqueta con el identificador de la región a la que pertenece. Posteriormente, para poder ordenarlas, el algoritmo RBS define la métrica 'rs' ('*rule significance*') como muestra la expresión 2.9.

$$rs(\, r_i \,) = \begin{cases} -1 - sop(\, r_i \,) \cdot conf\,(\, r_i \,) & \Leftrightarrow r_i \in \text{REG-1} \\ 1 + sop(\, r_i \,) \cdot conf\,(\, r_i \,) & \Leftrightarrow r_i \in \text{REG-2} \\ sop(\, r_i \,) \cdot conf\,(\, r_i \,) & \Leftrightarrow r_i \in \text{REG-3} \\ 0 & \Leftrightarrow r_i \in \text{REG-0} \end{cases} \qquad \text{(Expresión 2.9)}$$

En su trabajo, Rabasa llama a estas regiones '*regiones de significancia*' por el significado o interpretación que deberá darse a las reglas según se encuentren en un área u otra. Esa interpretación es la siguiente:

- **<u>Región 1 (REG-1)</u>:** Reglas de descarte.
  Estas reglas tienen un alto soporte y baja confianza, son por tanto reglas que pueden ser empleadas para descartar ciertas situaciones debido a la baja probabilidad de ocurrencia del consecuente; en otras palabras, en los datos de entrenamiento, el consecuente inferido por estas reglas se da muy pocas veces (para el antecedente considerado) por lo que se puede asegurar con alta probabilidad de acierto que dicho consecuente no va a producirse.

- **<u>Región 2 (REG-2)</u>:** Reglas directas.
  En la región 'REG-2' las reglas tienen tanto un alto soporte como una alta confianza, es decir, son confiables y de aplicación directa, así pues, se trata de reglas que proporcionan información fiable acerca del comportamiento de los datos de entrenamiento con los que se ha generado el modelo.

- **<u>Región 3 (REG-3)</u>:** Reglas a observar.
  Las reglas de la región 'REG-3' son aquellas que tienen un soporte muy bajo, se debe entender que cualquier conclusión que pudiera extraerse de ellas no deberá considerarse segura ya que no hay suficientes ejemplos en el conjunto de datos inicial que cumplan el antecedente, por lo que no se podrán extraer conclusiones estadísticamente fiables. No obstante, en ciertos ámbitos de estudio (como por ejemplo, en medicina, las denominadas enfermedades '*raras*') interesa conocer la existencia de dichas reglas para hacer con ellas análisis más pormenorizados guiados por un experto (o grupo de expertos).

- **<u>Región 0 (REG-0)</u>:** Reglas descartables.
  Por último, las reglas de la región 'REG-0' tienen niveles de soporte y confianza intermedios. Son estas las que el algoritmo RBS elimina para reducir el conjunto de reglas 'R' proporcionado como entrada del algoritmo, dando lugar al conjunto reducido 'Rr'.

Finalmente, el método define y calcula una métrica más denominada 'ACI' ('*Attribute Correlation Index*' o '*Índice de Correlación de Atributos*') que permite ponderar de manera global cómo están de correlacionados los atributos del antecedente del sistema de reglas con el atributo consecuente, por tanto, el 'ACI' es una métrica aplicable al conjunto total de reglas reducido 'Rr' (no a cada regla). Esta métrica 'ACI' se calcula según se indica en la expresión 2.10.

$$ACI(\,Rr\,) = \frac{(\,|\,REG\text{-}1\,|\,+\,|\,REG\text{-}2\,|\,+\,|\,REG\text{-}3\,|\,)\;/\;|\,R\,|}{area(\,REG\text{-}1\,) + area(\,REG\text{-}2\,) + area(\,REG\text{-}3\,)} \qquad \text{(Expresión 2.10)}$$

A modo de resumen, puede verse en el algoritmo 2.2 cómo quedarían los pasos del método RBS de reducción de reglas.

Algoritmo 2.2: RBS (R: Conjunto de reglas)		
1.	Inicializar ejes Ec y Es	(expresión 2.5)
2.	Calcular ejes Eci, Ecs, Esi, Ess	(expresión 2.6)
3.	Ajustar ejes Eci, Ecs, Esi, Ess para reducir regiones	(expresión 2.8)
4.	Calcular región y 'rs' da cada regla	(expresión 2.9)
5.	Calcular ACI	(expresión 2.10)
6.	Rr = R - { REG-0 }	
7.	Devolver Rr	

De los 6 pasos que presenta el algoritmo RBS (sin considerar el último, '*Devolver*'), los más costosos, desde el punto de vista de la complejidad computacional, únicamente implican realizar unas iteraciones sobre todas las reglas del conjunto de entrada 'R', por lo que puede afirmarse que se trata de un algoritmo de complejidad lineal con respecto al tamaño de 'R'. Nótese que en ningún momento el algoritmo hace cálculo alguno con los atributos o la clase de las reglas, solamente opera con sus valores de soporte y confianza, por lo que el número de atributos o la cantidad de valores asociados a los mismos no influyen en absoluto en el coste computacional, por tanto, dicho coste vendrá dado por la expresión 2.11.

$$\Theta(|R|) \qquad \text{(Expresión 2.11)}$$

El algoritmo RBS, ha sido aplicado con éxito en marcos tan diferentes como los GIS y la Ingeniería Agrónoma o la Medicina. Así, por ejemplo, en (Zaragozí et. al 2011) RBS fue utilizado para modelar el cambio de uso de las parcelas, atendiendo a los patrones de uso, características demográficas y topográficas del terreno, a partir de los voluminosos y heterogéneos datos recogidos por un GIS.

En el ámbito agronómico, el algoritmo RBS se ha empleado para la detección de las características hídricas más importantes para el diseño óptimo de planes de riego en el cultivo de viñedos (Rabasa et. al 2011). En el ámbito de la Medicina, el algoritmo RBS se empleó en varios sistemas de apoyo al diagnóstico: en el caso de enfermedades tiroideas (Almiñana et. al 2008), o en la detección temprana de cáncer de mama (Almiñana et. al 2010), entre otros.

2.2. Introducción al problema y objetivos

El algoritmo CREA, que se presenta en esta Tesis, surge a raíz de la pretensión de disponer de un único método que pudiera generar, en un solo paso, un conjunto de reglas reducido aplicando el método RBS. Como se ha observado en la figura 2.4, para que este algoritmo pueda funcionar, necesita de un conjunto de reglas de clasificación que estén debidamente caracterizadas con sus valores de soporte y confianza. En la práctica, de lo que se dispone normalmente es de un conjunto de datos (no del conjunto de reglas) que, si se quiere procesar con el método RBS, se le debe aplicar en un paso previo algún método de generación de reglas que, además, proporcione también el

soporte y la confianza de cada una de ellas. Dado que el estándar de facto para realizar dicha tarea es el algoritmo ID3, se optó por desarrollar un sistema integrado capaz de ejecutar en un único proceso ambos métodos, incorporando los cálculos necesarios para que cada regla generada lleve asociado también sus valores de soporte y confianza.

Una de las características fundamentales del algoritmo diseñado es la incorporación de un índice (vector) que facilita la división del conjunto de ejemplos 'E' en los sucesivos subconjuntos 'E_i'. Esta característica proporciona un ahorro muy significativo de tiempo en el repetido cálculo de las ganancias de información que se debe realizar varias veces en cada nodo interno (no-hoja) del árbol. Como se demostrará en la sección 3.2, la complejidad computacional de este algoritmo para calcular la ganancia en un nodo viene dada por la expresión 2.12.

$$\Theta_{CREA}(G) = Max (N/V^{h-1} , V^2) \qquad \text{(Expresión 2.12)}$$

Donde 'N' es el número de ejemplos del conjunto de datos de entrada, 'V' es el número de valores que pueden tomar los distintos atributos, y 'h' es la altura del nodo en el que se están haciendo los cálculos dentro del árbol de decisión. Como puede observarse, esta complejidad es sensiblemente menor que la calculada en (Quinlan 1986) y que se indicó en la expresión 2.4.

$$\Theta(N \times C) \qquad \text{(Repetición de la expresión 2.4)}$$

Téngase en cuenta que en los problemas habituales de Minería de Datos, el número de registros, 'N', es siempre mucho mayor que el de los posibles valores, 'V', que puede tomar cada atributo. Con menor complejidad, CREA genera sistemas de reglas que, además, reduce y clasifica por significancia ('rs'). Para unos mismos índices de precisión en ambos algoritmos, ésta es una diferencia fundamental respecto de ID3 cuyos sistemas de reglas obedecen al estricto orden de creación (y posterior recorrido en profundidad) del propio árbol. Estas mejoras en el ordenamiento automático de las reglas generadas y reducción del tiempo de cómputo, hacen que el algoritmo CREA sea idóneo para aplicarlo en problemas predictivos donde haya fuertes restricciones temporales que exijan llegar a obtener sistemas de reglas de clasificación más manejables por el usuario final y de la forma más ágil posible.

A raíz de lo anteriormente expuesto, se plantea la hipótesis de que para determinar la complejidad de un sistema generador de reglas de clasificación, no sólo debe tenerse en cuenta el número de registros ('N') y de atributos o columnas ('C'), sino también, el número de posibles valores ('V') que puedan adquirir dichos atributos. Se comprueba que este parámetro es determinante en el tiempo empleado en la generación del árbol. La experiencia computacional presentada, pone de manifiesto la capacidad de respuesta del algoritmo CREA bajo diferentes condiciones de carga y determina su complejidad computacional mediante una función definida por tramos.

Se quiere abordar también la cuestión del diseño de una estructura de datos optimizada en la que almacenar el conjunto de datos inicial ‘E’ que facilite las operaciones de recorrido conteo y verificación de los diversos valores de dicho conjunto de datos para reducir los tiempos de procesamiento, si bien no es frecuente que en un trabajo teórico sobre diseño de algoritmos se llegue al nivel de cómo implementar las estructuras de datos de más bajo nivel, lo cierto es que no todas las estructuras de datos son igualmente eficientes y, aunque en muchas ocasiones elegir una u otra no proporciona mejoras apreciables teóricamente (al realizar el estudio de la complejidad asintótica del algoritmo), empíricamente se comprueba que los tiempos de proceso sí pueden ser muy diferentes según las estructuras de datos que se implementen. Por ejemplo, obsérvese la instrucción de la expresión 2.13.

Si (var-1 = var-2) Entonces … (Expresión 2.13)

Se trata de una instrucción elemental, disponible en cualquier lenguaje de programación, cuyo coste computacional, en un estudio asintótico de la complejidad temporal, se considerará constante ($\boldsymbol{\Theta}(1)$). En la práctica, si las variables ‘var-1’ y ‘var-2’ son de tipo numérico, efectivamente la comparación se ejecutará siempre en un tiempo fijo (idealmente un ciclo de reloj), independientemente del valor que puedan tener, ya que, internamente, la unidad aritmético-lógica del procesador de un computador dispone de mecanismos para hacer rápidamente cualquier operación elemental con valores numéricos (sumas, restas, comparaciones, etc...). En cambio, si esas mismas variables son de tipo cadena, la operación de comparación podrá necesitar más o menos tiempo dependiendo del tamaño y contenido de dichas cadenas. En el peor de los casos, serán necesarias tantas iteraciones como caracteres tenga la menor de ambas, ya que la comparación, internamente, se realizará carácter a carácter (véase la expresión 2.14).

$$\boldsymbol{\Theta}(\text{ ‘}\textit{Si var-1 = var-2 Entonces ...}\text{’ }) = \text{Min }(\ |\text{var-1}|\ ,\ |\text{var-2}|)$$

(Expresión 2.14)

Empíricamente, la diferencia entre las operaciones de comparar dos variables de tipo numérico o dos de tipo cadena de caracteres es totalmente inapreciable si se ejecuta una única comparación (o un grupo de ellas), pero si esa operación se encuentra dentro de una serie de bucles anidados que iteran muchas veces (del orden de millones de veces), entonces sí se apreciará una diferencia notable en los tiempos de proceso observados. Según (Wirth 1978), para elaborar programas no basta con diseñar algoritmos, sino que es igualmente importante diseñar adecuadamente las estructuras de datos que se van a utilizar para almacenar tanto los datos de entrada al programa como la información intermedia que se ha de procesar.

Así pues, para poder abordar soluciones reales y rigurosas a los temas planteados previamente, esta Tesis se ha marcado los objetivos generales que se describen a continuación:

Objetivo 1:

Plasmar un estado del arte con clasificación bibliográfica formal del papel que está desempeñando la Minería de Datos en la resolución de problemas predictivos de Inteligencia de Negocios.

Como ya se ha indicado, si bien el método propuesto es de aplicación en diversos ámbitos de estudio, en esta Tesis se va a abordar el análisis de un caso real de Inteligencia de Negocios con datos de recepción de pedidos de la empresa MTNG GLOBAL EXPERIENCE, S.L., del sector del calzado. Por tanto es preceptivo realizar un estudio previo del estado del arte sobre la aplicación de técnicas de Minería de Datos en el dominio empresarial.

Objetivo 2:

Diseñar e implementar un procedimiento, es decir, un algoritmo con sus correspondientes estructuras de datos, de generación y reducción de reglas de clasificación que produzca, en tiempos mínimos, sistemas de reglas precisas y ordenadas.

En el epígrafe anterior 2.2 (y subapartados 2.2.1 y 2.2.2) ha quedado patente la principal razón de ser de este estudio: el desarrollo de un método de generación de modelos basados en reglas de clasificación aptos para alimentar sistemas de apoyo a la toma de decisiones, bajo las restricciones de tiempo y espacio que ya se han mencionado. Este algoritmo, como ya se ha descrito, entre otras cosas, integrará el método de Quinlan para generación de árboles de decisión con el de Rabasa para reducción de conjuntos de reglas de clasificación.

Objetivo 3:

Realizar un estudio formal de la complejidad computacional del algoritmo propuesto y verificar que el resultado del mismo es coherente con las pruebas empíricas realizadas.

Se ha generado una serie de conjuntos de datos simulados con diferentes valores de ‘N’ (número de registros), ‘C’ (número de columnas) y ‘V’ (número de posibles valores) para verificar que en cada caso, bajo diferentes condiciones de carga, el tiempo de procesamiento empírico es compatible con el análisis teórico de la complejidad asintótica del algoritmo CREA. Adicionalmente, también se verifica dicha compatibilidad con los datos empresariales.

Objetivo 4:

Introducir un sistema de obtención de características más relevantes de entre las disponibles (‘*feature selection*’) utilizando la métrica ‘ACI’ proporcionada por el algoritmo RBS.

Dado que la métrica 'ACI' permite determinar el grado de correlación entre los atributos antecedentes de un conjunto de reglas y la clase consecuente, para dos conjuntos de reglas 'Rr-i' y 'Rr-j' generados a partir del mismo conjunto de datos 'E' tal que los atributos considerados para generar dichos conjuntos han sido '{A-i}' y '{A-j}' (con '{A-i}'≠'{A-j}'), se puede afirmar que si 'ACI(Rr-i)' > 'ACI(Rr-j)', la combinación de atributos '{A-i}' contiene las características más relevantes para inferir la clase consecuente. Utilizando técnicas de fuerza bruta (cuando el número total de atributos disponibles es pequeño) o técnicas genéticas (cuando dicho número es grande), se puede obtener un método de extracción de características relevantes.

Objetivo 5:

Comprobar si se cumple la siguiente hipótesis: Dados dos conjuntos de datos con igual número de ejemplos, si se cumple que la potencia 'V^C' coincide en ambos casos, los tiempos de ejecución del algoritmo CREA para dichas muestras coinciden o son muy parecidos.

Sobre el tamaño total de un problema, independientemente del número de registros que tenga el conjunto de datos a procesar, existe un número de reglas máximo teórico que se podrían generar, que se calcula como el número total de posibles combinaciones que pueden hacerse con los diferentes valores que toman cada una de las columnas del conjunto de datos. Es decir, el número máximo de reglas teóricas posible que podría llegar a generarse es 'V^C', que es la clave en la generación de reglas de clasificación. Este hecho sugiere que dados dos conjuntos de datos 'E_1' y 'E_2' caracterizados por los valores 'N_1', 'C_1' y 'V_1' para 'E_1' y 'N_2', 'C_2' y 'V_2' para 'E_2', si el número de registros de ambos conjuntos coincide, es decir, '$N_1 = N_2$', si además se cumple que '$V_1^{C1} = V_2^{C2}$', cabría esperar que el tiempo de cálculo sea muy similar, incluso idéntico, para ambos conjuntos de datos. Esta hipótesis se puede expresar más formalmente mediante la expresión 2.15, y esta Tesis se plantea como uno de sus objetivos verificar si se cumple o no.

$$N_1 = N_2 \ \wedge \ V_1^{C1} = V_2^{C2} \ \rightarrow \ \Theta(\ CREA(E_1)\ ) \cong \Theta(\ CREA(E_2)\ )$$

(Expresión 2.15)

UNIVERSITAS
Miguel
Hernández

Capítulo 3
Algoritmo propuesto y estudio de la complejidad computacional

3.1. Especificación formal del algoritmo de generación, reducción y ordenación de sistemas de reglas: CREA

En este capítulo se presenta el algoritmo CREA (‘*Classification Rules Extraction Algorithm*’) para generación, reducción y ordenación de reglas de clasificación, que combina en un único proceso el método ID3 de generación de reglas de clasificación (Quinlan 1979) y el método RBS de reducción y ordenación de reglas de clasificación (Rabasa 2009). El algoritmo desarrollado no es una mera conexión en serie de dos procesos que se ejecutan consecutivamente uno tras otro, sino que incorpora ciertas propiedades que mejoran notablemente el rendimiento computacional del proceso global, como se verá al final del capítulo en el epígrafe 3.3. A continuación, en este apartado, se aclaran una serie de conceptos generales y definiciones, y se presenta la notación que se va a emplear, para posteriormente explicar las principales características del algoritmo y el diseño de las estructuras de datos que se han utilizado en su implementación. Posteriormente se presentará en detalle el algoritmo CREA y se realizará un minucioso estudio de su complejidad computacional.

3.1.1. Conceptos generales y definiciones

La entrada del algoritmo es un conjunto de ejemplos, ‘E’, formado por datos discretos (o discretizados) con ‘C’ atributos o columnas, cada uno de los cuales puede tomar uno de entre ‘V’ posibles valores, y un total de ‘N’ registros o filas (los ‘ejemplos’), tal y

como se muestra en figura 3.1. En principio, cada columna del conjunto de datos podría tener asociado un número diferente de valores posibles, pero para simplificar la notación del estudio, y sin pérdida de generalidad, se va a suponer que todos los atributos pueden tomar el mismo número de valores, 'V'.

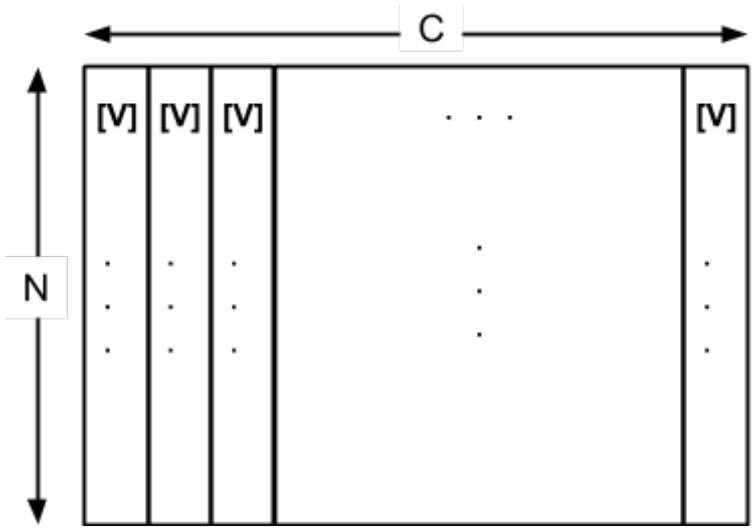


Figura 3.1: Conjunto de datos de entrada con 'N' tuplas, 'C' atributos y 'V' posibles valores

En principio, cualquiera de las columnas podría ser empleada como valor de clase o consecuente, pero en cualquier momento, y sin perjuicio del resultado final, se pueden reordenar las columnas del conjunto 'E' de forma que la variable de clase quede ubicada a la derecha del conjunto de datos de entrada. En adelante se asumirá que las columnas 1 a 'C-1' son las que constituyen el antecedente y la columna 'C-ésima' será el atributo consecuente de las reglas a generar.

Sea 'R' el conjunto total de las reglas presentes (al menos una vez) en la muestra, y que se corresponden con un determinado registro 'i', bajo la forma: *antecedente* → *consecuente*, cada una de las reglas 'r_i' tendrá la forma:

$$r_i: a_1 = v_{i,1} , a_2 = v_{i,2} , \ldots , a_{C-1} = v_{i,C-1} \rightarrow a_C = v_{i,C} \qquad \text{(Expresión 3.1)}$$

Nótese que el número total de reglas presentes, '|R|', nunca podrá ser mayor que el número total de registros del conjunto de datos objeto de estudio, 'N'; ni tampoco podrá ser mayor que el número total de combinaciones posibles que se pueden realizar con los 'V' valores cada una de las 'C' columnas de datos ('V^C'), por tanto:

$$|R| \leq \text{Min}\,(N , V^C) \qquad \text{(Expresión 3.2)}$$

Para problemas de gran tamaño en 'V' y sobre todo en 'C', la exponencial 'V^C' supera rápidamente a 'N' y por tanto resulta que:

$$|R| \leq N \leq V^C \qquad \text{(Expresión 3.3)}$$

Aunque para valores pequeños de ‘C’ y ‘V’, y dado que es habitual en problemas de Minería de Datos, que el volumen de ejemplos a considerar (‘N’) sea muy grande, se cumplirá:

$$|R| \leq V^C \leq N \qquad \text{(Expresión 3.4)}$$

La figura 3.2 muestra el tipo de árbol generado durante el proceso de obtención de las reglas, donde ‘V’ es el número máximo de hijos de cada nodo del árbol, y ‘C’, que es el número de atributos, también representa la altura máxima del mismo árbol, por lo que habrá ‘C+1’ niveles que se numeran desde 0 (nivel del nodo raíz), hasta ‘C’, nivel máximo de los nodos finales de R (nodos hoja).

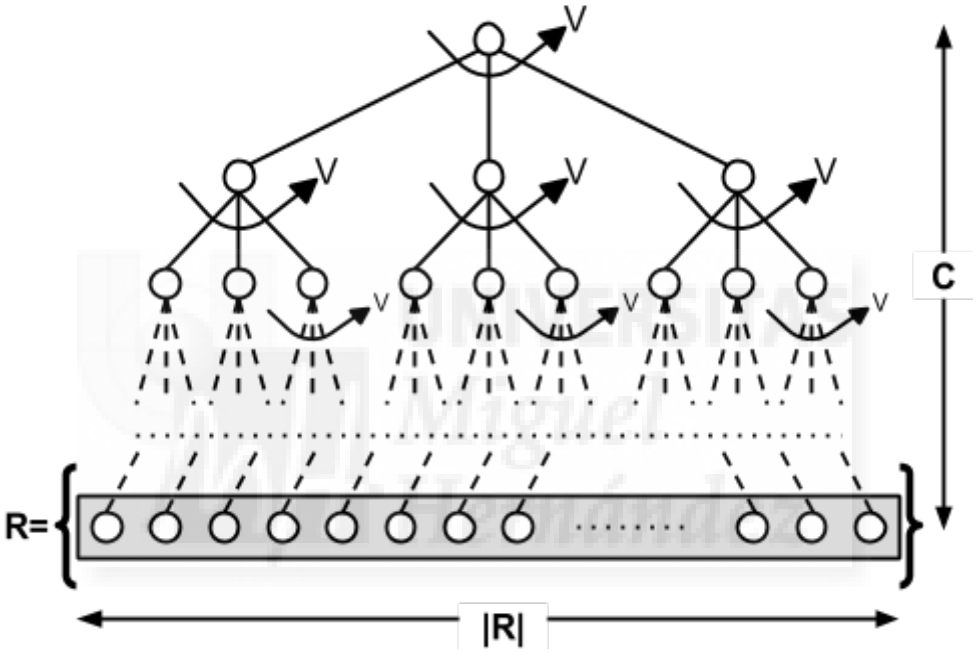


Figura 3.2: Esquema de expansión del Árbol de Decisión

Nótese que no todos los nodos de ‘R’ se expandirán completamente hasta llegar a un nivel ‘C’ ya que, dependiendo del conjunto de datos que se esté analizando en cada caso, puede ocurrir que una regla alcance el consecuente en un nivel inferior a ‘C’. Por ejemplo, dentro de ‘R’ puede haber reglas como la regla ‘r_i’, vista en la expresión 3.1, en la que todos sus atributos toman algún valor, y otras reglas como ‘r_j’:

$$r_j: a_1 = v_{j,1} , a_2 = v_{j,2} , \ldots , a_{C-1} = X \rightarrow a_C = v_{j,C} \qquad \text{(Expresión 3.5)}$$

Nótese que la regla ‘r_i’ (expresión 3.1) está totalmente expandida (todos los atributos de la regla toman algún valor), en cambio, para la regla ‘r_j’ no ha sido necesario determinar el valor del atributo ‘a_{C-1}’ para inferir el consecuente (entiéndase que la etiqueta ‘X’ significa que dicho atributo no tiene ningún valor asignado); por tanto, ‘r_j’ es una regla de ‘R’ que no ha necesitado expandir todos sus atributos. En otras palabras, todas las reglas de ‘R’ totalmente expandidas tendrán una profundidad igual a ‘C’ dentro del

árbol de decisión de la figura 3.2, mientras que las reglas no totalmente expandidas tendrán una profundidad menor que ‘C’. Se puede considerar como un caso pesimista (más tiempo de proceso) aquel en el que todos los nodos de ‘R’ tienen profundidad ‘C’.

Por otra parte, cada nodo del árbol va a contener un índice con los registros del conjunto de datos con los que es compatible. Dicho índice consiste en un vector de enteros encargado de referenciar todos los ítems del conjunto ‘E’ asociados con el nodo en cuestión (ver figura 3.3).

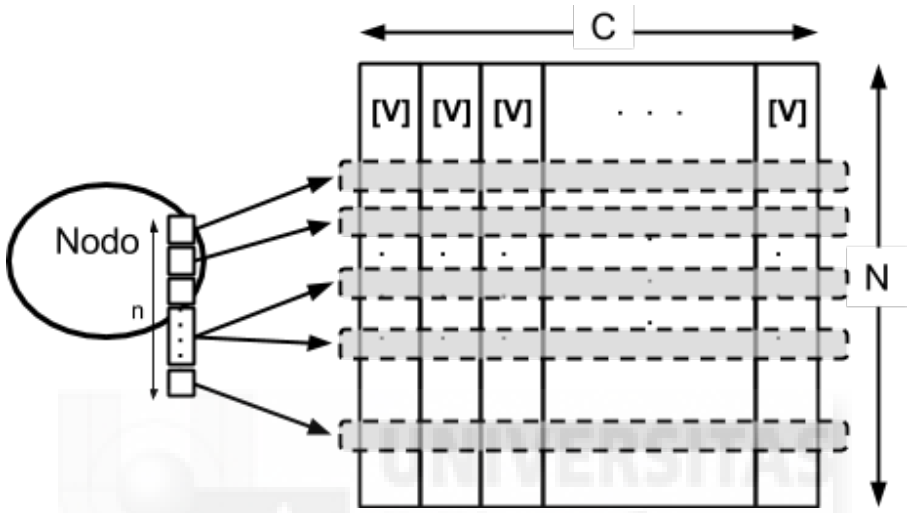


Figura 3.3: Índice de un nodo genérico referenciando ítems del conjunto de datos

Inicialmente, el nodo raíz, ‘0’, que aún no ha sido expandido por ninguno de los ‘C’ atributos posibles, es compatible con los ‘N’ ejemplos del conjunto de datos ‘E’, por lo que tendrá asociado un índice (vector) de tamaño ‘N’ donde cada elemento de dicho índice referenciará a cada una de las filas del conjunto ‘E’. Según se van expandiendo nuevos nodos, el índice se va fragmentando para cada uno de ellos (ver figura 3.4), y en cada caso el índice del nodo padre deja de ser necesario, por lo que el tamaño total de todos los índices de todos los nodos del árbol, pendientes de expandir, siempre será ‘N’ (o menor que ‘N’ si alguno de los nodos ya es un nodo final perteneciente a ‘R’). Es preciso notar que el mantenimiento de este índice tiene una complejidad temporal lineal y únicamente se necesita disponer de un vector de ‘N’ enteros para todo el proceso.

Considérese ahora que se va a emplear el concepto de ganancia de información (en adelante ‘*ganancia*’) como criterio para determinar, dado un nodo, cuál es el atributo (columna) por el que se debe seguir expandiendo (véase, paso 6 del algoritmo 2.1 ‘TDIDT’). Para su cálculo se necesita determinar cuántas filas del conjunto original de datos son compatibles con el nodo a expandir. El citado índice agiliza dicho cálculo ya que gracias a él se conoce el número de registros que son compatibles con un nodo en concreto, por lo que se reduce el número de iteraciones. De no utilizar este índice, u otro mecanismo similar, habría que calcular cada ganancia recorriendo siempre los ‘N’ registros del conjunto de datos (en todos los nodos). Por tanto, con este índice, tan solo hay que recorrer en cada nodo las filas que le son compatibles. Como se puede observar

en la figura 3.4, esa cantidad de filas se va reduciendo con la profundidad del árbol. Suponiendo que cada vez que se expande un nodo en sus 'V' hijos, estos reducen en '1/V' el tamaño del índice que heredan de su nodo padre, en tal caso, cada nodo tendría un índice de tamaño 'N/V^h', siendo 'h' el nivel del árbol en el que se encuentra el nodo. En cualquier caso, sea cual sea la partición que se haga cada vez que un nodo se expande, el tamaño acumulado de todos los nodos de un mismo nivel (expansión en anchura), será 'N' (o menor que 'N' si alguno de los nodos ya es nodo-hoja perteneciente a 'R'). La incorporación de este índice supone una notable mejora en el tiempo computacional de la ganancia, respecto al diseño clásico del algoritmo ID3.

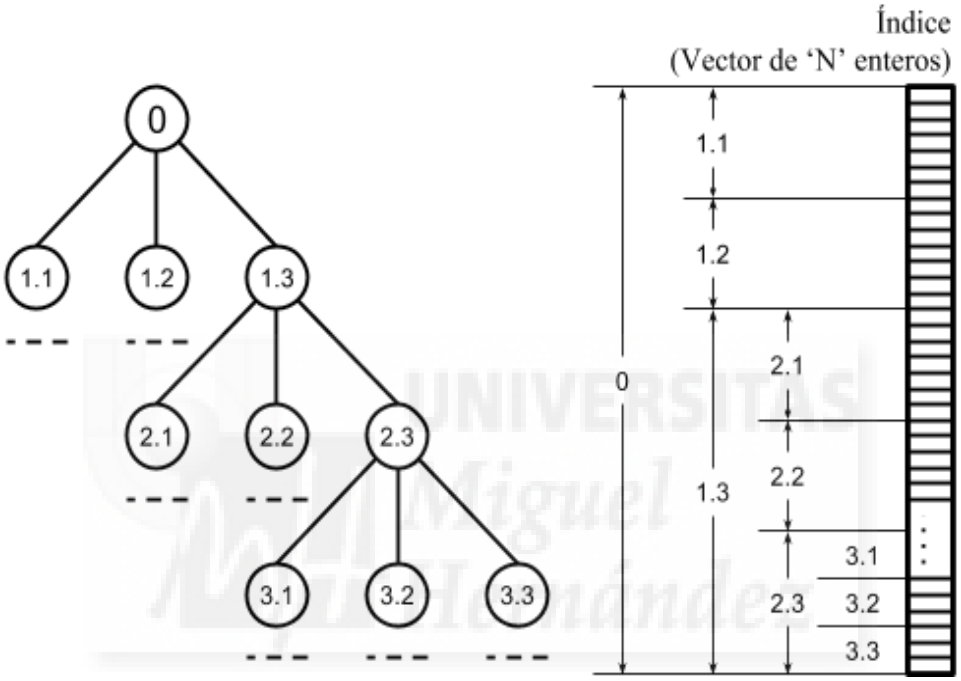


Figura 3.4: Árbol de Decisión e índice de nodos asociado

En este caso, el algoritmo CREA incorpora el método RBS para reducción y ordenación de reglas, cuyo objetivo es la clasificación óptima de las mismas por regiones de significancia, y que los propios autores sometieron a un riguroso análisis en (Almiñana et al. 2012), donde la métrica propuesta, 'rs' ('*rule significance*'), es comparada cualitativa y cuantitativamente con las medidas más utilizadas expuestas y evaluadas en profundidad en (Tan et al. 2004). Por tanto, se asume que el conjunto resultante de reglas, tras aplicar el algoritmo CREA, ya está debidamente clasificado, en el mismo momento en que se genera, a diferencia de lo que ocurre con otros árboles tipo ID3, que necesitan un post-procesado ad-hoc para ello.

3.1.2. Principales características del algoritmo CREA

CREA es un algoritmo de generación y ordenación de reglas de clasificación para datos discretos que incorpora un conjunto de mejoras sobre otros algoritmos del mismo tipo, que lo hacen sensiblemente más rápido. Una primera característica de este algoritmo es

que se trata de un proceso iterativo en vez de recursivo (como suele ser habitual en este tipo de problemas, véase algoritmo 2.2, TDIDT). Este planteamiento ya supone una cierta ventaja, debido a que las soluciones recursivas consumen más recursos al requerir reservar nueva memoria para las variables del proceso que se llama a sí mismo de forma recurrente. Además, estas llamadas recursivas requieren más tiempo de procesamiento que el diseño iterativo equivalente. Aunque los algoritmos recursivos normalmente son más fáciles de entender e implementar (suelen tener un diseño más elegante), por razones de eficiencia, se ha optado por un diseño e implementación iterativos.

Otra peculiaridad del método CREA es que no construye físicamente en memoria un árbol de decisión propiamente dicho, donde cada regla sería un recorrido desde el nodo raíz hasta alguna de sus hojas (aunque conceptualmente dicho árbol se utilizará para explicar detalles del algoritmo). Por el contrario, se van construyendo las reglas iterativamente a partir del conjunto de datos de entrada y se van almacenando en una lista de '*nodos vivos*'; seleccionado un nodo de esa lista para expandir. Sus posibles nodos hijos copian la información contenida en el nodo padre e incorporan la nueva información generada en la expansión. Dicha información consistirá en un par '<*atributo,valor*>' donde '*atributo*' será el campo seleccionado como mejor según el criterio de ganancia, y '*valor*' será alguno de los valores que dicho atributo puede tomar. En ese punto, el nodo padre es eliminado de la lista de '*nodos vivos*' y los nodos hijos, con esa nueva información, se insertan en ella para volver a ser expandidos en una iteración posterior. Cuando un nodo es expandido por última vez por el atributo consecuente, los nodos hijos resultantes ya son reglas definitivas y pasan a almacenarse en el conjunto de reglas 'R'.

También es una característica propia del método CREA que para un mismo antecedente, el algoritmo puede generar todas las reglas posibles (una por cada posible valor de la variable consecuente) determinando cuál de ellas es la mejor categorización, e incluso permite hacer un '*ranking*' con dichas reglas para el antecedente considerado. Nótese que para esas reglas, el valor de '*soporte*' (probabilidad del antecedente) será exactamente el mismo, variando únicamente la '*confianza*' de cada una de ellas en función del consecuente que en cada caso corresponda. Como curiosidad, comentar que cuando estas reglas se representan gráficamente en un plano '*conf*|*sop*' (ver figuras 2.4 y 2.5) aparecen distribuidas en una misma línea horizontal.

Por otra parte, la principal aportación del algoritmo CREA es que mantiene en cada nodo de la lista de '*nodos vivos*', un índice de forma que cada regla en construcción apunta en todo momento a todos aquellos registros (ítems) del conjunto inicial de datos compatibles con la propia regla. Como se ha explicado en el epígrafe 3.1.1, la construcción y mantenimiento de este índice, permite optimizar sensiblemente el cálculo de la ganancia, ya que cada vez que se hace esta operación (y se hace muchas veces) no hay que recorrer todo el conjunto de datos inicial, sino solamente aquella parte de los datos referenciados por el índice de la regla considerada en cada caso.

Otra característica importante del método CREA es que incorpora un mecanismo de reducción y clasificación de las reglas generadas que permite eliminar las menos importantes (estadísticamente) y clasificar las consideradas más relevantes según los criterios de significancia del método RBS (Rabasa 2009) que ya se explicó detalladamente en el epígrafe 2.1.2.

3.2. Estructuras de datos

Como se introdujo en el epígrafe 2.2, las estructuras de datos subyacentes en todo algoritmo, así como los métodos de actualización y consulta que las manipulan, pueden ser determinantes a la hora de conseguir mejoras en los tiempos de ejecución. En esta Tesis se ha dado especial importancia a la forma en que los datos son manipulados y, por tanto, a las estructuras en los que han de ser almacenados. Por un lado, se ha diseñado una estructura '*Columna*' para guardar los datos de los ejemplos de entrenamiento, en la que se almacenarán los valores de cada uno de los atributos del conjunto de datos de entrada 'E'. Por otra parte, se ha diseñado una estructura '*Regla*' sobre la que ir construyendo las reglas en sucesivas iteraciones, a la vez que se mantiene el índice que hace referencia a los ítems del conjunto de datos de entrada que son compatibles con la regla que en cada caso se esté considerando.

3.2.1. La estructura '*Columna*'

La estructura '*Columna*' representa uno de los atributos del conjunto de datos de entrada 'E' (véase figura 3.5). Esta estructura contiene tanto los datos como los metadatos del atributo 'A' que almacena. Es decir, si el conjunto 'E' tiene 'N' ítems y el atributo 'A' puede tomar 'V' valores, la estructura '*Columna*' contendrá una cadena para guardar el nombre, 'A', del atributo, un vector de 'V' cadenas de caracteres para guardar los valores posibles del atributo y un entero para guardar el dato 'N' (número de datos almacenados). Hasta aquí, toda esa información constituye los metadatos del atributo.

Para almacenar los datos propiamente dichos se crea un vector de números de tamaño 'N'. En cada posición de este vector se almacenará un valor entero que, en la práctica, representa una posición del vector de los 'V' valores de los metadatos. En otras palabras, si la posición i-ésima del vector de datos toma el valor '1', el valor nominal correspondiente a dicha posición es el que hay almacenado en la posición '1' en el vector de 'V' valores de los metadatos. Se ha optado por esta forma de almacenar la información porque, como se justificó en el epígrafe 2.2, las operaciones de comparación, que se realizan de forma intensiva en el algoritmo CREA (y en muchos otros algoritmos de Minería de Datos), son computacionalmente más ágiles si se realizan con variables numéricas en lugar de cadenas de caracteres.

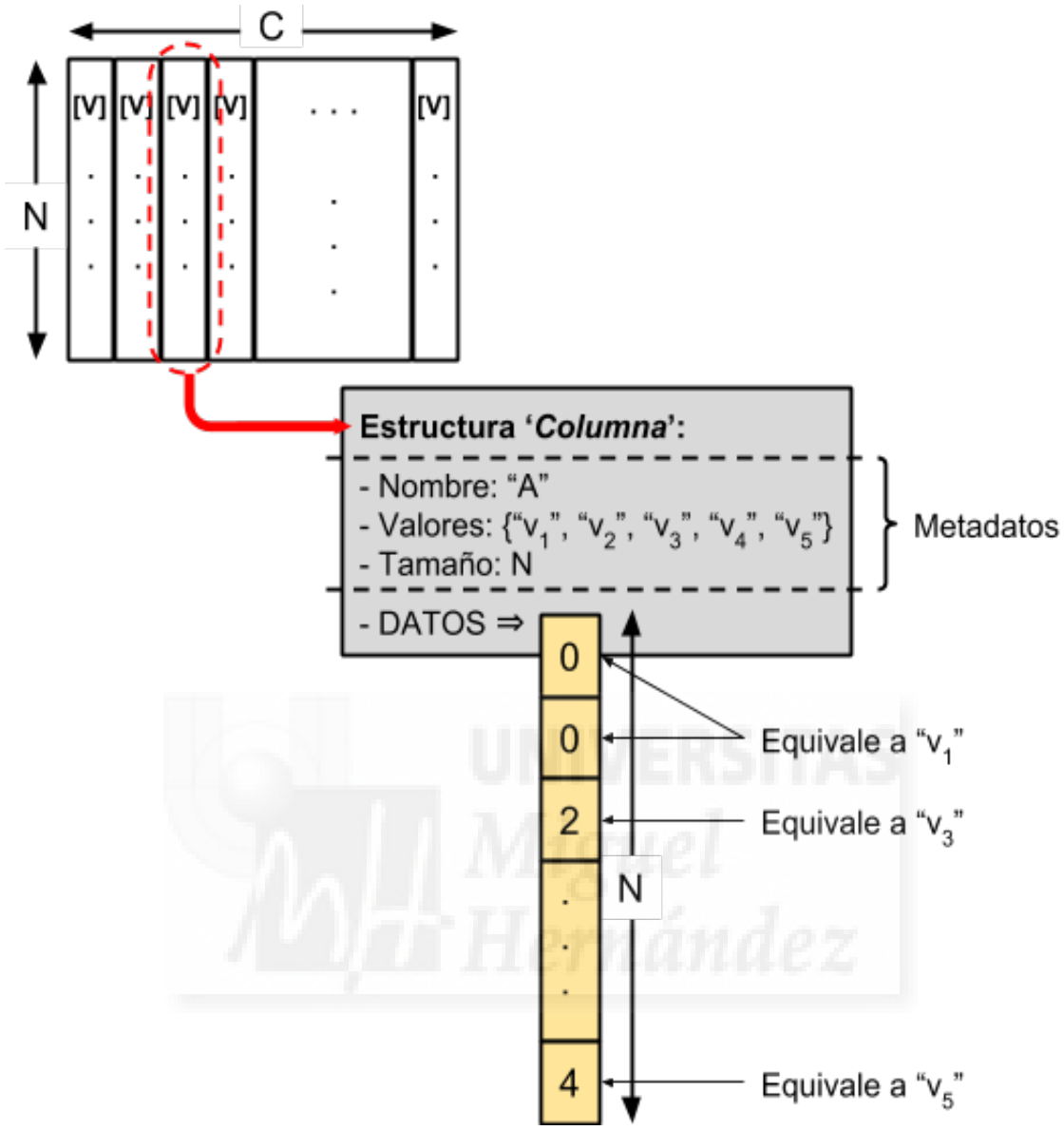


Figura 3.5: Ejemplo de columna que puede tomar 5 valores posibles ('V')

Vista la representación en memoria de una columna de datos, el conjunto total 'E' estará formado por un grupo de 'C' columnas dispuestas en un vector como el de la figura 3.6. Se opta por trabajar con referencias (punteros) para agilizar el paso de parámetros entre funciones, por tanto, ese conjunto de columnas se implementa en realidad como un vector de punteros a columnas. Esta forma de representar el conjunto de datos 'E' tiene una segunda ventaja, y es que con los datos dispuestos de esta forma en memoria, resulta sencillo crear subconjuntos de datos (por ejemplo 'E´' o 'E´´') con un número de columnas ('C´' o 'C´´') menor que las del conjunto de datos inicial. Este hecho permite programar con relativa facilidad algoritmos en los que hacer múltiples ejecuciones del algoritmo CREA (o cualquier otro algoritmo que trabaje con matrices de datos discretos) considerando en cada caso un subconjunto de columnas del conjunto original 'E'. Este mecanismo resulta muy práctico para diseñar procesos, bien de fuerza bruta o bien genéticos, que busquen optimizar la métrica 'ACI' (véase método RBS en el

epígrafe 2.1.2) para diferentes combinaciones de atributos antecedentes. También resulta práctico para, una vez cargados los datos en memoria, cambiar la columna consecuente en diferentes ejecuciones del proceso CREA (u otro método de clasificación).

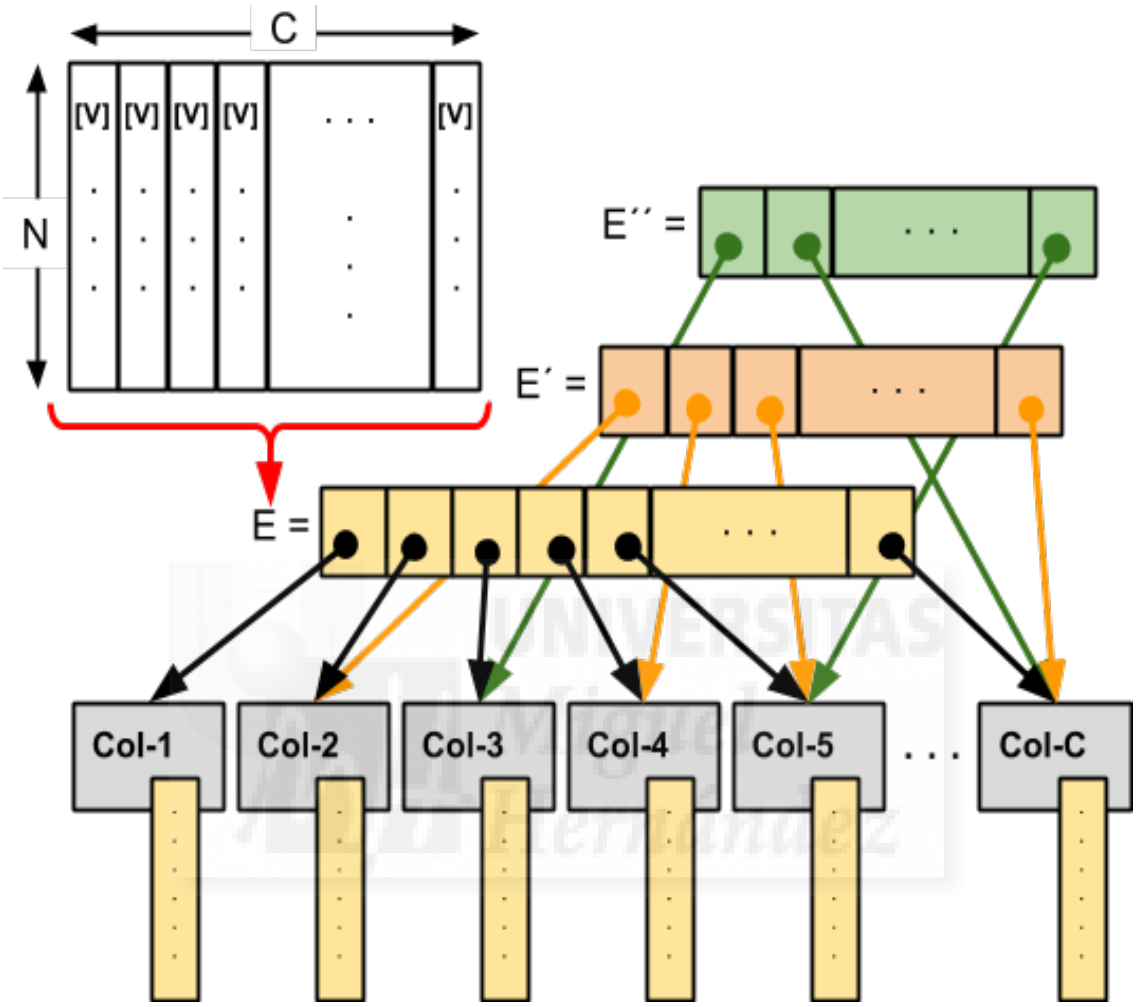


Figura 3.6: Almacenamiento de datos en memoria con vectores de 'Columnas'

3.2.2. La estructura '*Regla*'

La estructura '*Regla*' se emplea para construir las reglas de clasificación. Como ya se ha mencionado, cada estructura '*Regla*', no solamente debe almacenar la información relativa a la regla que iterativamente se va formando, sino que además debe disponer de un índice que registre los ejemplos del conjunto de entrenamiento, 'E', que en cada momento son compatibles con dicha regla. En el epígrafe 3.1.1 (véase figura 3.3) ya se introdujo este hecho para justificar la mejora computacional que supone disponer de ese índice; ahora se va a explicar con mayor detalle cómo se ha diseñado la estructura '*Regla*' para su implementación y uso dentro del algoritmo CREA. Nótese que, al tratarse de una estructura que debe funcionar de forma genérica con cualquier conjunto de datos (independientemente del número de filas y columnas que tenga o del número

de posibles valores que pueda tomar cada atributo) todos los vectores que contiene la estructura son vectores dinámicos, es decir, no se crean inicialmente como vectores con un tamaño fijo, sino que al ejecutarse el proceso se determina cuánta memoria es necesaria y se reserva en ese momento. En el algoritmo 3.1 se muestra el pseudocódigo con el diseño de la estructura '*Regla*' que, como puede verse, contiene información diversa que puede agruparse en varias categorías.

Algoritmo 3.1: Estructura REGLA

```
Estructura REGLA
// ÍNDICE PARA EL CÁLCULO DE LA GANANCIA
1.    Entero: nIndice          // Tamaño del índice
2.    Vector Entero: INDICE    // Identifica los registros compatibles con la regla

//REFERENCIA AL ORIGEN DE DATOS: PARTE ESTÁTICA Y DE SOLO LECTURA
3.    Entero: N                // nº de ejemplos (filas) del origen de datos
4.    Entero: C                // nº de atributos (columnas) del origen de datos
5.    Matriz Entero:&ANT       // Referencia a los atributos antecedentes, matriz[C-1 x N]
6.    Vector Entero:&CON       // Referencia al atributo consecuente, vector[N]

// INFORMACIÓN DE LA REGLA
7.    Entero: nivel            // Nivel del nodo → nº de atributos ya expandidos
8.    Bool: esHoja             // Identifica si el nodo se ha terminado de expandir
9.    Vector Entero: ATR       // Contiene IDs de los atributos seleccionados
10.   Vector Entero: VAL       // Contiene los valores de los atributos seleccionados

// ATRIBUTOS PARA EL MÉTODO RBS
11.   Real: conf               // Confianza de la regla
12.   Real: sop                // Soporte de la regla
13.   Entero: reg              // Región de la regla (0, 1, 2, 3)
14.   Real: rs                 // Métrica 'rule significance' (importancia de la regla)
Fin Estructura
```

En la parte '*Índice para calcular la ganancia*' (algoritmo 3.1, líneas 1-2), la variable 'INDICE' es el vector que permitirá hacer referencia a los registros del conjunto de datos compatibles con la regla y, 'nIndice' indicará el tamaño de dicho vector.

La categoría '*Referencia al origen de datos*' (algoritmo 3.1, líneas 3-6) sirve para poder acceder a los datos de entrada del algoritmo CREA. Nótese que esta parte de la estructura se declara explícitamente como '*estática*', eso quiere decir que inicializando estos valores una única vez en la estructura '*Regla*' al arrancar el proceso, cualquier instancia que se haga de dicha estructura durante la ejecución del algoritmo dispondrá automáticamente de la misma información para poder acceder a los datos. El carácter '&' quiere expresar que la matriz 'ANT' es una referencia (puntero) a los datos de los atributos antecedentes, por tanto se trata de una matriz de 'N' filas y 'C-1' columnas. Del mismo modo 'CON' es una referencia al vector de datos del atributo consecuente de tamaño 'N'. Otra característica importante de esta parte de la estructura es que es solamente de lectura, es decir, se empleará para consultar los datos de entrada, nunca para modificarlos. La forma de acceder a las filas del conjunto de datos que son compatibles con una regla se hace a través de 'ANT' y 'CON' utilizando el vector

'INDICE' como indirección. En el algoritmo 3.2 se muestra un ejemplo de cómo hacer uso de estos elementos para acceder a los registros deseados.

Algoritmo 3.2: Ejemplo del uso de 'INDICE' como indirección para 'ANT' y 'CON'

```
1.  REGLA: r1                    // 'r1' es una estructura de tipo 'Regla'
2.  Entero: fila, columna
3.  ...
4.  Para fila = 1 Hasta r1.nIndice
5.      Para columna = 1 Hasta ri.C - 1
6.          Procesar_Valor_Antecedente( r1.ANT[ columna ][ INDICE[ fila ] ] )
7.      Fin Para
8.      Procesas_Valor_Consecuente( r1.CON[ INDICE[ fila ] ] )
9.  Fin Para
```

Nótese como en las líneas 6 y 8 del algoritmo 3.2 (resaltado en rojo) se utilizan los valores del vector 'INDICE' (indexados por la variable 'fila') para indexar los registros de datos con 'ANT' y 'CON' respectivamente.

La '*Información de la regla*' (algoritmo 3.1, líneas 7-10) contiene los datos de la regla de clasificación propiamente dicha. A medida que esta se va construyendo, los vectores 'ATR' y 'VAL' (de tamaño 'C') contendrán los pares '<atributo,valor>' conforme se calculen, 'nivel' determinará el número de atributos que ya se han expandido (se corresponde con el nivel del árbol de decisión), y 'esHoja' es un dato booleano que determina cuándo la regla ha terminado de expandirse. En otras palabras, cuando el parámetro 'esHoja' toma el valor '*verdadero*' se entenderá que la regla ya es una regla final y por tanto, forma parte del conjunto 'R'.

Finalmente, los '*Atributos para el método RBS*' (algoritmo 3.1, líneas 11-14) se utilizan para caracterizar la regla con los parámetros del método RBS. Cuando la regla ha expandido el último atributo antecedente, la variable 'nIndice' contiene el número de ítems que son compatibles con ella en ese momento, por tanto, en ese punto se puede calcular el soporte de la regla trivialmente según la expresión 3.1.

$$sop = \frac{nIndice}{N} \qquad \text{(Expresión 3.1)}$$

De igual forma, cuando una regla ha expandido también su atributo consecuente y ya se encuentra totalmente formada (pertenece al conjunto 'R'), el atributo 'nIndice' estará actualizado con el número de registros del conjunto de datos que coinciden con la regla. Su confianza se calcula entonces con la expresión 3.2.

$$conf = \frac{nIndice}{sop \cdot N} \qquad \text{(Expresión 3.2)}$$

3.3. Pseudocódigo

A continuación se va a detallar el diseño del algoritmo CREA que se presenta en esta Tesis. Primeramente se muestra el esquema general del proceso iterativo de construcción de las reglas de clasificación y seguidamente se especifican los cálculos necesarios para aplicar el método RBS de reducción y ordenación de reglas, así como el detalle de varias funciones a las que se llama desde el procedimiento general. El algoritmo 3.3 muestra el pseudocódigo de dicho proceso principal (el algoritmo CREA propiamente dicho). Se observa que en él aparecen ciertas líneas sombreadas; se han representado con sombreado naranja los trozos de código correspondiente a los cálculos del método RBS que se detallan más adelante. Se han sombreado de amarillo las llamadas a las funciones que se usan en el procedimiento principal y cuyo pseudocódigo también se detalla más adelante, en este mismo apartado.

Nótese que las variables ‘r’ y ‘rAux’ que se utilizan en el algoritmo son estructuras de tipo ‘*Regla*’, que incluyen toda la información que se especifica en el algoritmo 3.1. Por tanto, la primera línea del algoritmo CREA (‘*r ← Regla vacía*’), crea una estructura de tipo ‘*Regla*’, ‘r’, sin ningún atributo expandido y con un índice de tamaño ‘N’ (‘*nIndice=N*’) que referenciará a todas las filas de los datos de entrada ‘E’. El resto de atributos de la estructura no requieren inicializarse con ningún valor en concreto, salvo los atributos de referencia al origen de datos (estáticos y de solo lectura) que se supone previamente inicializados y referenciando convenientemente al origen de datos. Inicialmente esta regla es la única contenida en la lista ‘LNV’ (‘*lista de nodos vivos*’, línea 2). La lista ‘LR’ (‘*lista de Reglas*’, línea 3) es también una lista para almacenar estructuras de tipo ‘*Regla*’ que se usará más adelante para guardar dichas reglas cuando estén completamente formadas.

El bucle principal del algoritmo (líneas 5-24) itera indefinidamente hasta que la lista ‘LNV’ queda vacía. El propósito general de este bucle es sacar nodos de ‘LNV’ (línea 6), identificar el mejor atributo, ‘Atr’, por el que expandir el nodo extraído (línea 7), realizar dicha expansión añadiendo el atributo ‘Atr’ elegido al nodo (línea 8), duplicar dicho nodo una vez por cada posible valor ‘V1’ de ‘Atr’ (bucle de las línea 9-23) y almacenar los nodos expandidos, bien otra vez en la lista ‘LNV’ (línea 21) (si aún no son nodos finales) o bien en la lista ‘LR’ si ya se ha expandido el atributo consecuente (línea 13 o línea 17).

Se observa que hay dos puntos en el algoritmo CREA (líneas 11 y 15) donde se verifica si el nodo es o no final (atributo ‘esHoja = TRUE’), ya que hay dos formas en que un nodo puede convertirse en una regla final y hay que tratarlas de forma diferente. La condición '*Final-1*' (línea 15) ocurre cuando todos los ejemplos apuntados por el índice de la regla pertenecen a la misma clase; en este caso se genera una única regla final cuyo consecuente será dicha clase. La condición '*Final-2*' (línea 15) se da cuando un nodo ya ha expandido los 'C-1' primeros atributos (el antecedente completo de la regla)

y únicamente queda tratar el atributo consecuente (la variable de clase); en este caso se generan tantas reglas como posibles valores pueda tomar el consecuente, dichos valores serán los consecuentes de las reglas generadas.

Algoritmo 3.3: CREA

Entrada:

E [C-1 x N]: Matriz de datos con 'C-1' columnas (atributos) y 'N' filas (registros)
CAtr: Atributo/Columna consecuente

Salida:

LR: Lista de reglas (reducida y ordenada según criterio RBS)
ACI: Índice de correlación de atributos

Utiliza:

LNV: Lista de reglas (lista de nodos vivos, contiene reglas no completadas)
r, rAux: estructuras de tipo 'Regla'
Atr, AtrAux: Entero (atributos/columna)
V1, V2: Entero (posibles valores de alguna de las columna)

```
INICIO
 1.   r   ← Regla vacía // inicializa la regla del nodo raíz
 2.   LNV ← r           // insertar 'R' en 'LNV'
 3.   LR  ← Lista vacía
 4.   [INICIALIZACIÓN VARIABLES RBS]
 5.   Mientras (LNV no Vacía) Hacer
 6.         r   ← SacarNodo(LNV) // elimina de 'LNV' el nodo extraído
 7.         Atr ← GetBestAtr(r)
 8.         r   ← R + Atr        // incluye el 'R' el nuevo atributo
 9.         Para V1 = 0 Hasta |Atr|
10.               rAux ← GetNextRule(r, Atr, V1)
11.               Si (rAux es regla Final-1) Entonces
12.                     V2 ← Valor ∈ CAtr : rAux es Final-1
13.                     LR ← GetNextRule(rAux, CAtr, V2)
14.                     [CÁLCULOS-1 PARA EJES RBS]
15.               Sino-Si (rAux es regla Final-2) Entonces
16.                     Para V2 = 0 Hasta |CAtr|
17.                           LR ← GetNextRule(rAux, CAtr, V2)
18.                           [CÁLCULOS-1 PARA EJES RBS]
19.                     Fin Para
20.               Sino
21.                     LNV ← rAux
22.               Fin Si
23.         Fin Para
24.   Fin Mientras
25.   [CÁLCULOS-2 PARA EJES RBS]
26.   Para cada r ∈ LR
27.         Determinar la región y 'rs' de r y calcular ACI según RBS
28.         Si (Región de r es 'REG-0') Entonces
29.               Eliminar r de LR
30.         Fin Si
31.   Fin para
FIN
```

Finalizado el bucle principal del algoritmo CREA, se dispone del conjunto completo de reglas de clasificación caracterizadas cada una de ellas con un valor de soporte y confianza. En este punto se deben concretar los cálculos para los ejes del método RBS que delimitan las regiones 'REG-0', 'REG-1', 'REG-2' y 'REG-3'. Definidas estas áreas, para todas las reglas generadas, se identifica la región a la que pertenece cada una de ellas y se calcula su valor de 'rs' (métrica '*rule significance*') que permite crear un ranking con dichas reglas por orden de importancia. Finalmente, queda eliminar las reglas pertenecientes a la región 'REG-0' y calcular la métrica 'ACI' del conjunto de reglas final, 'LR' (líneas 26 a 31). A continuación se pasa a describir con más detalle cada una de las operaciones que han quedado resaltadas en el algoritmo 3.3 (sombreadas en naranja y amarillo).

El algoritmo 3.4, '*Inicialización de variables RBS*' (se corresponde con la línea 4 del algoritmo 3.3), establece los valores de los ejes 'Ec' y 'Es'. Se inicializan a cero cuatro variables de tipo entero que realizarán el recuento de las reglas finales según se ubiquen, bien a la izquierda o a la derecha del eje 'Ec', o bien por encima o por debajo del eje 'Es'. También se inicializan a cero otras cuatro variables de tipo real que se utilizarán para hacer los sumatorios de soporte y confianza de las reglas, dependiendo, como en el caso anterior, de en qué lado de los ejes 'Ec' o 'Es' queden dichas reglas. Más adelante (ver algoritmo 3.6) se utilizarán estas ocho variables para determinar, calculando promedios, los ejes que delimitan las regiones del método RBS.

Algoritmo 3.4: Inicialización variables RBS

```
// Ejes iniciales del método RBS
Ec = 1.0 / |Vc|// Nº valores del consecuente
Es = 1.0 / (|V1|·|V2|·…·|Vc-1|) // Nº combinaciones antecedente

// Variables de tipo entero para conteo de reglas
EciCont = 0 // nº de reglas a la izquierda de 'Ec'
EcsCont = 0 // nº de reglas a la derecha de 'Ec'
EsiCont = 0 // nº de reglas por debajo de 'Es'
EssCont = 0 // nº de reglas por encima de 'Es'

// Variables de tipo real para calcular los ejes que delimitan
// las regiones del método RBS
Eci = 0.0   // Eje 'conf' inferior
Ecs = 0.0   // Eje 'conf' superior
Esi = 0.0   // Eje 'sop' inferior
Ess = 0.0   // Eje 'sop' superior
```

El algoritmo 3.5, '*Cálculos-1 para ejes RBS*', se ejecuta cada vez que se genera una regla final (líneas 14 y 18 del algoritmo 3.3: CREA). En ese punto, se verifica el soporte y la confianza de la regla generada, y según corresponda, se actualizan las variables para los sumatorios y contadores del método RBS inicializadas en el algoritmo 3.4.

Algoritmo 3.5: Cálculos-1 para ejes RBS

```
// Recuento de las reglas que están por debajo o por
// encima del eje 'Es', y sumatorio de 'conf' y 'sop'
Si (Soporte(rAux)<Es) Entonces
      Esi = Esi + Soporte(rAux)
      EsiCont = EsiCont + 1
Sino
      Ess = Ess + Soporte(rAux)
      EssCont = EssCont + 1
Fin si

// Recuento de las reglas que están a la izquierda o a la
// derecha del eje 'Ec', y sumatorio de 'conf' y 'sop'
Si (Confianza(rAux)<Ec) Entonces
      Eci = Eci + Confianza(rAux)
      EciCont = EciCont + 1
Sino
      Ecs = Ecs + Confianza(rAux)
      EcsCont = EcsCont + 1
Fin si
```

En el algoritmo 3.6, '*Cálculos-2 para ejes RBS*', se corresponde con la línea 25 del algoritmo 3.3 (CREA). Ahora simplemente se calculan los ejes 'Eci', 'Ecs', 'Esi' y 'Ess' como promedio de las confianzas y soportes inferiores y superiores de las reglas finales generadas.

Algoritmo 3.6: Cálculos-2 para ejes RBS

```
Si EciCont > 0 Entonces// Se verifica que los contadores
      Eci = Eci/EciCont     // sean mayores que cero para
Sino// evitar una división por cero
      Eci = 0
Fin si
Si EcsCont > 0 Entonces
      Ecs = Ecs/EcsCont
Sino
      Ecs = Ec
Fin si
Si EsiCont > 0 Entonces
      Esi = Esi/EsiCont
Sino
      Esi = 0
Fin si
Si EssCont > 0 Entonces
      Ess = Ess/EssCont
Sino
      Ess = Es
Fin si
```

Nótese, que estos tres últimos algoritmos (3.4, 3.5 y 3.6) no introducen ningún bucle o iteración adicional en el algoritmo principal. Se trata de operaciones secuenciales,

integradas en el algoritmo CREA, bien al principio (algoritmo 3.4), o bien dentro del bucle principal (algoritmos 3.5 y 3.6). Por lo tanto, desde un punto de vista asintótico, no afectan a la complejidad computacional del proceso.

Para terminar con las operaciones relativas al método RBS, solamente queda identificar la región a la que pertenece cada regla final contenida en la lista 'LR', así como su valor de '*rs*'. Clasificadas las reglas por regiones, se recalculan los ejes 'Eci', 'Ecs', 'Esi' y 'Ess', y se calcula la métrica 'ACI'. Estas operaciones son las descritas en el algoritmo 3.7.

Algoritmo 3.7: Determinar regiones, significancia de cada regla y ACI según RBS

```
numReglasTotal = |LR|  //LR: Lista de Reglas finales

confMax = 0.0
confMin = 1.0
sopMax  = 0.0
sopMin  = 1.0

Para cada r ∈ LR
      Según método RBS:
       - Identificar región de r: REG-0, REG-1, REG-2 o REG-3
       - Calcular la métrica 'rs' de r según región

      Si (r ∈ REG-0) Entonces
            Eliminar r de LR
      Fin si

      Si (r ∈ REG-1 Y confMax < r.conf) Entonces
            confMax = r.conf
      Fin si
      Si (r ∈ REG-2 Y confMin > r.conf) Entonces
            confMin = r.conf
      Fin si
      Si (r ∈ REG-3 Y sopMax < r.sop) Entonces
            sopMax = r.sop
      Fin si
      Si (r ∈ {REG-1 ∪ REG-2} Y sopMin > r.sop) Entonces
            sopMin = r.sop
      Fin si

      Eci = confMax
      Ecs = confMin
      Esi = sopMax
      Ess = sopMin
Fin Para

ACI = (|REG-1|+|REG-2|+|REG-3|)/ numReglasTotal /
(Area(REG-1)+Area(REG-2)+Area(REG-3))
```

En esta ocasión, el algoritmo 3.7 sí es otro bucle, pero se encuentra fuera del bucle principal de CREA por lo que su coste computacional se sumaría al coste computacional de dicho bucle principal. Desde la perspectiva del cálculo asintótico de la complejidad, el coste computacional de este segundo bucle será despreciable frente al primero, ya que este segundo bucle es lineal con respecto al número de reglas generadas, '|R|'. Es decir, tendrá un peso mucho menor que el bucle principal.

La función 'GetBestAtr' (algoritmo 3.8, llamada en la línea 7 del algoritmo 3.3) identifica, para una regla no final dada, el atributo que maximiza la ganancia de información de entre los que aún están pendientes de ser expandidos. Por tanto, está estableciendo el atributo por el que se deberá seguir expandiendo la regla considerada. Esta función, a su vez, llama a la función 'Gain' cuyo funcionamiento se describe en el algoritmo 3.9.

Algoritmo 3.8: Función GetBestAtr (obtener mejor atributo)

Entrada:

r: Regla no final

Salida:

bestAtr: mejor atributo para expandir la regla r

Usa:

atr: entero (atributo de la regla)
gain: real (ganancia de información)
bestGain: real (ganancia de información)

```
INICIO
 1.   Si (r.nivel ≥ r.C-2) Entonces
 2.         Devolver [el único 'atr' que queda por expandir]
 3.   Fin si
 4.
 5.   bestAtr = -1
 6.   bestGain = -∞
 7.
 8.   Para atr = 0 Hasta C-1.
 9.         Si (atr no ha sido seleccionado previamente) Entonces
10.               gain = Gain(r, atr)
11.               Si (gain > bestGain) Entonces
12.                     bestGain = gain
13.                     bestAtr  = atr
14.               Fin si
15.         Fin si
16.   Fin para
17.   Devolver bestAtr
FIN
```

Nótese, que en la primera línea de 'GetBestAtr' se verifica si el nivel actual de la regla es igual (o mayor) que el número total de atributos menos 2 ('C-2'). Es decir, se

comprueba si la regla ha llegado a su nivel antepenúltimo. En tal caso no es necesario calcular la ganancia, ya que en ese nivel queda un único atributo por ser expandido, por lo que no se hace necesario determinar si hay otro mejor (pues no hay otro candidato disponible).

El cálculo de la ganancia viene dado por la función 'Gain' descrita en el algoritmo 3.9. Es conveniente tener presentes las expresiones 2.1 y 2.2 que definen dicho cálculo y que se introdujeron en el epígrafe 2.1.1.

$$I(E) = -\sum_{i=1}^{n} p_i \log_2(p_i) \quad \text{(Repetición expresión 2.1)}$$

$$G(A) = I(E) - \sum_{i=1}^{v} p(E_i)\, I(E_i) \quad \text{(Repetición expresión 2.2)}$$

Para el cálculo de las probabilidades y los sumatorios de las expresiones 2.1 y 2.2 se necesita una serie de contadores que son inicializados en las líneas 1-6 del algoritmo 3.9. En las líneas 7-13 se hace el recuento de los valores que toma el atributo considerado 'atr', para el cual se está calculando la ganancia (vector '*contadorVA*'); así como los valores que toma el atributo consecuente 'CAtr' globalmente (vector '*contadorVC*'), y desglosados por valores del antecedente (matriz '*contadorVC2*'). El bucle de las líneas 15-20 realiza el sumatorio que calcula el primer término de la ganancia ('I(E)', ver expresión 2.1). Finalmente, el bucle de las líneas 21-30 completa el cálculo añadiendo el sumatorio de los términos '$I(E_i)$' (ver expresión 2.2).

Algoritmo 3.9: Función Gain (ganancia de información)

Entrada:

r: Regla no final
atr: Entero (atributo considerado para calcular su ganancia de información)
E [C-1 x N]: Matriz Entero (datos con 'C-1' columnas y 'N' filas)
CAtr: Entero (atributo consecuente)

Salida:

gain: Real (ganancia de información)

Usa:

VAtr: entero (valor del atributo 'atr' del antecedente)
VCAtr: entero (valor del atributo/columna consecuente 'CAtr')
contVA[|atr|]: Vector Entero (conteo de valores de 'atr')
contVC[|CAtr|]: Vector Entero (conteo de valores de 'CAtr')
contVC2: Array [|atr| x |CAtr|] : Matriz Entero (conteo de valores de 'CAtr')
pr: real (auxiliar para cálculo de probabilidad)
ent: real (auxiliares para cálculo de entropía)

(Sigue →)

Algoritmo 3.9: Función Gain (ganancia de información) *(Continuación)*

```
INICIO
 1.   Para i=0 Hasta |CAtr|          // INICIALIZAR CONTADORES
 2.        contVA[i] = contVC[i] = 0
 3.        Para j=0 Hasta |atr|
 4.             contV[i][j] = 0
 5.        Fin para
 6.   Fin para
 7.   Para i=0 Hasta r.nIndice       // CONTEO DE VALORES
 8.        VAtr  = E[atr][ r.INDICE[i] ]
 9.        VCAtr = CAtr[ r.INDICE[i] ]
10.        contVA [ VAtr  ] = contVA [ VAtr  ] + 1
11.        contVC [ VCAtr ] = contCC [ VCAtr ] + 1
12.        contVC2[ VAtr  ][VCAtr] = contVC2[ VAtr ][VCAtr] + 1
13.   Fin para
14.   gain = 0
15.   Para i=0 Hasta |CAtr|  // Cálculo 1º ganancia: expresión 2.1
16.        Si (contVC[i]>0) Entonces// se considera que: 0*log(0)  ⇒0
17.             pr = contVC[i] / r.nIndice
18.             gain = gain - ( pr * log2(pr) )
19.        Fin si
20.   Fin para
21.   Para j=0 Hasta |atr|   // Cálculo 2º ganancia: expresión 2.2
22.        ent = 0
23.        Para i=0 Hasta |CAtr|
24.             Si (contV[j][i] > 0) Entonces// 0*log(0)  ⇒0
25.                  pr  = contV[j][i]/contVA[j]
26.                  ent = ent - ( pr / log2(pr) )
27.             Fin si
28.        Fin para
29.        gain = gain - ( ent * contVA[j] / r.nIndice )
30.   Fin para
31.   Devolver gain
FIN
```

Por último, la función ‘GetNextRule’ (líneas 10, 13 y 17 del algoritmo 3.3), se encarga de expandir las reglas que se van extrayendo de la ‘*lista de nodos vivos*’, actualizando el índice que apunta al subconjunto de ejemplos compatibles con la regla de que se trate en cada caso (ver algoritmo 3.10). En esta función, primero se calcula el tamaño que deberá tener el nuevo índice de la regla expandida ‘r2’ (líneas 2-6), se reserva memoria para dicho índice (líneas 8 y 9), y se actualiza con los datos correspondientes de la regla ‘r’ original cuyos valores para el atributo de entrada, ‘atr’, coinciden con el valor ‘val’ para ese atributo (líneas 10-18). Se aprovecha ese mismo bucle para determinar si el atributo consecuente de los ítems de ‘r2’ toma el mismo valor en todos los casos (necesario para saber si la regla ya es final según la condición ‘Final-1’ que se describió en el algoritmo 3.3, línea 11). Una vez actualizado el índice de la nueva regla ‘r2’, se copian en ella los valores que tenía la regla original ‘r’ (líneas 19-22), se actualiza la nueva regla con su nuevo nivel y con el nuevo atributo considerado y su valor (líneas 23-25).

Algoritmo 3.10: función GetNextRule (obtener siguiente regla)

Entrada:

r: Regla no final
atr: Atributo por el que expandir la regla
val: Valor asignado al atributo 'atr'
E [C x N]: Matriz de datos con 'C' columnas (atributos) y 'N' filas (registros)

Salida:

r2: Regla expandida

Usa:

cont: entero (contador)
i, j: entero (indice)
mismoV: Booleano (determinar si los valores del consecuente son iguales)
VC1: entero (valor del consecuente para la primera regla indexada)

```
INICIO
 1.   cont = 0, j = 0, mismoV = TRUE
 2.   Para i = 0 Hasta r.nIndice
 3.         Si (E[atr][ r.INDICE[i] ] = val) Entonces
 4.               cont = cont + 1
 5.               Si (cont=1) Entonces VC1 = E[ C ][ r.INDICE[i] ]
 6.         Fin si
 7.   Fin para
 8.   r2.INDICE ← Vector de tamaño 'cont' // reservar memoria
 9.   r2.nIndice = cont
10.   Para i = Hasta r.nIndice
11.         Si (E[atr][ r.INDICE[i] ] = val) Entonces
12.               r2.INDICE[j] = r.INDICE[i]
13.               Si(E[C][r2.INDICE[j]] ≠ VC1) Entonces
14.                     mismoV = FALSO
15.               Fin si
16.               j = j + 1
17.         Fin si
18.   Fin para
19.   Para i = 0 Hasta r.nivel
20.         r2.ATR[i] = r.ATR[i]
21.         r2.VAL[i] = r.VAL[i]
22.   Fin para
23.   r2.nivel = r.nivel + 1
24.   r2.ATR[ r2.nivel ] = atr
25.   r2.VAL[ r2.nivel ] = val
26.   Si (r2.esHoja) Entonces
27.         r2.conf = r2.nIndex / (r2.sop * N)
28.   Sino - Si (mismoV O r2.nivel = C-1) Entonces
29.         r2.esHoja = TRUE
30.         r2.sop = r2.nIndex / N
31.         Si (mismoV) Entonces
32.               r2.conf = 1 // confianza del 100%
33.         Fin si
34.   Fin si
35.   Devolver r2
FIN
```

Si el nodo ya es un '*nodo hoja*' ('esHoja = TRUE'), es decir, si se identificó como hoja en una iteración anterior, solamente queda calcular su confianza según la expresión 3.2 (líneas 27-28). Si la regla no es '*nodo hoja*' aún (esto es, no fue caracterizada como tal en una iteración anterior), hay que comprobar si en este punto debe pasar a ser marcada como '*nodo hoja*'. En caso de que el índice del nodo apunte a ítems, todos ellos con idéntico valor para el atributo consecuente ('mismoV = TRUE'), o si ya se ha expandido el último atributo antecedente ('nivel = C-1'), se caracteriza el nodo como '*nodo hoja*' y se calcula su soporte según la expresión 3.1 (líneas 30 y 31). Si además, se da la condición de que todos los ítems apuntados con el índice tienen el mismo valor, entonces también se puede determinar que la confianza de la regla es 1.0 (líneas 32-34).

Recuérdese la condición denominada 'Final-1' de la función CREA (algoritmo 3.3, línea 11) para identificar un nodo hoja cuyo índice apunta a ejemplos del conjunto de datos con idéntico consecuente. Dado que la función 'GetNextRule' devuelve la regla expandida 'r2', para identificar en la función CREA si la regla es final, se verificará si se cumple la condición 'r2.esHoja = TRUE'. Adicionalmente, la regla 'r2' cumplirá la condición 'Final-1' si además tiene una confianza del cien por cien ('r2.conf = 1').

3.4. Complejidad computacional

Como ya se indicó al principio de este capítulo, en el epígrafe 3.1.1, para el estudio teórico de la complejidad computacional del método CREA, se va a suponer que todas las columnas del conjunto de datos de ejemplo 'E' toman el mismo número de valores 'V'. Esta asunción permite simplificar la notación y sistematizar mejor el estudio. En un caso hipotético (más realista) en el que hubiera diferentes valores de 'V' para cada columna del conjunto 'E', tal que 'Vmax' y 'Vmin' sean las cantidades de valores máxima y mínima que toman los atributos de 'E', los resultados de los estudios teóricos para 'V=Vmax' y 'V=Vmin' serán, respectivamente, cotas superior e inferior de la complejidad del caso hipotético planteado. También se va a asumir que cada vez que un nodo se expande, sus 'V' nodos hijos se reparten equitativamente los ejemplos que les son compatibles. En otras palabras, si el índice del nodo padre tiene tamaño 'n', los índices de los nodos hijos tendrán tamaño 'n/V', por tanto, en el nivel 'h' del árbol, el número de ejemplos para un nodo dado será 'N/V^h'. Como se justifica más adelante, esta suposición no afectará al cálculo global de la complejidad computacional.

Primeramente, se va a abordar el estudio de la función para el cálculo de la ganancia de información ('*Gain*') descrita en el algoritmo 3.8, que presenta 4 bucles principales consecutivos, algunos con otro bucle anidado. Asintóticamente, el coste computacional de esta función estará determinado por el más complejo de dichos bucles. Del análisis de estos bucles se desprende que esta función depende fundamentalmente del número de filas a recorrer 'N/V^h' ('nIndex') y del producto '|CAtr| x |atr|', es decir, de 'V^2' (ya que

se asumió que '|CAtr|' = '|atr|' = 'V'). De lo anterior se desprende que la complejidad asintótica de esta función es la indicada en la expresión 3.3.

$$\boldsymbol{\Theta}(\text{Gain}) = \text{Max}(N/V^h , V^2) \qquad \text{(Expresión 3.3)}$$

Expresada como función por partes, en la expresión 3.4.

$$\boldsymbol{\Theta}(\text{Gain}) = \begin{cases} N/V^h & \Leftrightarrow N/V^h > V^2 \Leftrightarrow N > V^{h+2} \\ V^2 & \Leftrightarrow N/V^h \leq V^2 \Leftrightarrow N \leq V^{h+2} \end{cases} \qquad \text{(Expresión 3.4)}$$

En un problema habitual de Minería de Datos, el número de ejemplos a considerar ('N') suele ser bastante mayor que 'V^2'. De este hecho y de las expresiones anteriores se desprende que para el nodo raíz y los nodos pertenecientes a los primeros niveles del árbol, la complejidad computacional de la ganancia vendrá dada por '$\boldsymbol{\Theta}(\text{Gain})=N/V^h$', mientras que para los nodos de los niveles medios y bajos del árbol la complejidad será '$\boldsymbol{\Theta}(\text{Gain})=V^2$'.

Para ilustrar este hecho considérese el siguiente ejemplo: Sea 'E' un conjunto de datos con 2 millones de registros ('N'), 30 columnas ('C') y 5 valores por columna ('V'); la complejidad de la ganancia en cada nivel del árbol es la siguiente:

- Nivel 00: $\boldsymbol{\Theta}(\text{Gain}) = N/V^0 = 2 \times 10^6$
- Nivel 01: $\boldsymbol{\Theta}(\text{Gain}) = N/V^1 = 4 \times 10^5$
- Nivel 02: $\boldsymbol{\Theta}(\text{Gain}) = N/V^2 = 8 \times 10^4$
- Nivel 03: $\boldsymbol{\Theta}(\text{Gain}) = N/V^3 = 16 \times 10^3$
- Nivel 04: $\boldsymbol{\Theta}(\text{Gain}) = N/V^4 = 32 \times 10^2$
- Nivel 05: $\boldsymbol{\Theta}(\text{Gain}) = N/V^5 = 640$
- Nivel 06: $\boldsymbol{\Theta}(\text{Gain}) = N/V^6 = 128$
- Nivel 07: $\boldsymbol{\Theta}(\text{Gain}) = N/V^7 = 25{,}6 (\sim V^2) \Rightarrow$ (punto de estabilización)
- Nivel 08: $\boldsymbol{\Theta}(\text{Gain}) = V^2 = 25$

  . . .
- Nivel 19: $\boldsymbol{\Theta}(\text{Gain}) = V^2 = 25$
- Nivel 30: $\boldsymbol{\Theta}(\text{Gain}) = V^2 = 25$

A lo largo del resto del estudio, se denotará como '$\boldsymbol{\Theta}(G_h)$' la complejidad de la ganancia en un nodo de nivel 'h' en el árbol.

La ganancia debe calcularse varias veces en cada nodo (función 'GetBestAtr', algoritmo 3.8), una por cada posible atributo por el que se podría expandir dicho nodo. En el nivel 0 (nodo raíz) los atributos candidatos son 'C-1' (la columna 'C-ésima' sería la columna consecuente que no es considerada a efectos de expansión de un nodo hasta el final, para determinar las reglas de 'R'). Por tanto, la complejidad en el nivel 0 será '(C-1) x $\boldsymbol{\Theta}(G_0)$'. En el segundo nivel hay 'V' nodos en los que la ganancia se calcula 'C-2' veces, es decir 'V x (C-2) x $\boldsymbol{\Theta}(G_2)$'. En general, la complejidad en cada nivel del árbol puede ser expresada de la siguiente manera:

- Θ(nivel 0) = (C - 1) x $\Theta(G_0)$
- Θ(nivel 1) = V x (C - 2) x $\Theta(G_1)$
- Θ(nivel 2) = V^2 x (C - 3) x $\Theta(G_2)$
- Θ(nivel 3) = V^3 x (C - 4) x $\Theta(G_3)$

  . . .
- Θ(nivel h) = V^h x (C-h-1) x $\Theta(G_h)$

Nótese que las expresiones anteriores se derivan del hecho de haber asumido que el número de registros compatibles de un nodo viene dado por la división 'N/V^h'. En la práctica no suele ser así, ya que dependerá en cada caso de los datos de entrada. En realidad, el coste computacional real del cálculo de la ganancia para un nodo dado podría variar con respecto al que se está considerado teóricamente, pero ocurre que, globalmente, para todos los nodos de un mismo nivel del árbol, las iteraciones necesarias para calcular la ganancia son siempre proporcionales a 'N' (en un caso pesimista, ya que algún nodo puede haberse convertido en regla final antes del nivel 'C'). Si denotamos por '$\Theta(G_h(i))$' el coste computacional de procesar la ganancia del nodo 'i-ésimo' del nivel 'h', se cumplirá que:

$$\Theta(G_h(1)) + \ldots + \Theta(G_h(V^h)) \cong V^h \times \Theta(G_h) \qquad \text{(Expresión 3.5)}$$

Es decir, en el caso pesimista, la suma de los costes reales de procesar todos los nodos de un nivel se aproximará mucho al valor teórico que se ha calculado asumiendo que el número de ítems compatibles con cada nodo es 'N/V^h'.

El nivel máximo del árbol está determinado por el número de columnas, el nivel 'C' es el de los nodos finales, cada uno de los cuales es una regla de 'R'. En ese nivel no hay que expandir más el árbol. En el nivel 'C-1' ya está expandido el último atributo del antecedente de la regla, y nada más queda generar una regla por cada posible consecuente. Por ello tampoco procede calcular la ganancia. Tampoco es necesario este cálculo en el nivel 'C-2' porque esos nodos tienen pendiente de expandir un único atributo del antecedente, es decir, al haber sólo una opción no hay que determinar cuál es la mejor (no hay varias alternativas). En consecuencia, el cálculo de la ganancia debe hacerse en todos los nodos de los niveles 0 a 'C-3', por lo que la complejidad del algoritmo CREA queda como indica la expresión 3.6:

$$\Theta(\text{CREA}) = \sum_{h=0}^{C-3} V^h \times (C - h - 1) \times \Theta(G_h) \qquad \text{(Expresión 3.6)}$$

Donde 'V^h' es el número máximo de nodos que puede haber en el nivel 'h' del árbol. Ahora bien, teniendo en cuenta que esa cantidad de nodos está acotada superiormente por la expresión 3.2.

$$|R| \leq \text{Min}(N, V^C) \qquad \text{(Repetición expresión 3.2)}$$

Por otra parte, en un determinado nivel 'h' no puede haber más nodos que en el 'C-ésimo' nivel de los nodos finales (reglas), '|R|', ni puede haber más reglas que ejemplos, 'N', o posibles combinaciones de los diferentes valores de los atributos, 'V^C', por tanto:

$$V^h \leq |R| \leq \text{Min}(N, V^C) \qquad \text{(Expresión 3.7)}$$

Por lo que al calcular esta complejidad, para aquellos valores de 'h' para los que '$V^h > N$', habría que sustituir 'V^h' por 'N'. De esta condición y de la fórmula para la complejidad de la ganancia '$\boldsymbol{\Theta}(G_n)$' (expresión 3.6) resulta la siguiente expresión para el cálculo de la complejidad:

$$\boldsymbol{\Theta}(\text{CREA}) = \begin{cases} \sum_h N \times (C\text{-}h\text{-}1) \times V^2 & \Leftrightarrow N < V^C \\ \sum_h V^h \times (C\text{-}h\text{-}1) \times V^2 & \Leftrightarrow V^C \leq N \leq V^{C+2} \\ \sum_h V^h \times (C\text{-}h\text{-}1) \times N/V^h & \Leftrightarrow N > V^{C+2} \end{cases} \qquad \text{(Expresión 3.8)}$$

De donde se concluye que la complejidad temporal del algoritmo CREA definida por intervalos es:

$$\boldsymbol{\Theta}(\text{CREA}) = \begin{cases} N \times V^2 \times \sum_h (C\text{-}h\text{-}1) & \Leftrightarrow N < V^C \\ \sum_h V^{h+2} \times (C\text{-}h\text{-}1) & \Leftrightarrow V^C \leq N \leq V^{C+2} \\ N \times \sum_h (C\text{-}h\text{-}1) & \Leftrightarrow N > V^{C+2} \end{cases} \qquad \text{(Expresión 3.9)}$$

Según proponen los autores Gilles Brassard y Paul Bratley (Brassard and Bratley 2000), el estudio de la complejidad temporal de un algoritmo debe abordarse desde un punto de vista asintótico y así se ha considerado en todo el análisis realizado en este epígrafe. A partir de la expresión 3.9, es posible obtener otra más simplificada pero igualmente coherente con el planteamiento asintótico del análisis.

En primer lugar debe tenerse en cuenta que el subíndice 'h' que aparece en los sumatorios de la expresión 3.9 ('$\sum_h$') varía entre 0 y 'C-3' (véase expresión 3.6), por tanto, atendiendo a la expresión para calcular sumas de series aritméticas, se tiene que:

$$\sum_h (C - h - 1) = \frac{C^2 - C - 2}{2} \qquad \text{(Expresión 3.10)}$$

Nótese que ahora, en los tramos primero y tercero de la expresión 3.9, puede sustituirse todo el sumatorio por 'C^2', asintóticamente, el término más influyente de la expresión 3.10. Por otra parte, el segundo tramo de la expresión 3.9 consiste en un sumatorio de exponenciales:

$$\sum_h V^{h+2} \text{ x } (C\text{-}h\text{-}1) = (C\text{-}1)\cdot V^2 + (C\text{-}2)\cdot V^3 + \ldots + 2\cdot V^{C-1} \qquad \text{(Expresión 3.11)}$$

Desde un punto de vista asintótico, el término más influyente de este sumatorio es el último 'V^{C-1}'. Con estas consideraciones, la expresión 3.9 se puede reformular, finalmente, del siguiente modo:

$$\boldsymbol{\Theta}(\text{CREA}) = \begin{cases} V^2 \text{ x } C^2 \text{ x } N & \Leftrightarrow \; N < V^C \\ V^{C-1} & \Leftrightarrow \; V^C \leq N \leq V^{C+2} \\ C^2 \text{ x } N & \Leftrightarrow \; N > V^{C+2} \end{cases} \qquad \text{(Expresión 3.12)}$$

Nótese que en un problema real de Minería de Datos, lo habitual es que 'N' sea mucho mayor que 'C^2' y que 'V^2', por lo que se puede concluir que el coste computacional del algoritmo CREA queda definido en tres tramos, en el primero de ellos la complejidad será lineal con respecto a 'N' y con pendiente '$C^2 \cdot V^2$'. En el segundo tramo la complejidad es exponencial de orden 'V^{C-1}'. Y en el tercer tramo vuelve a ser lineal con pendiente 'C^2', menor que en el primer tramo.

Finalmente, queda puntualizar que para el cálculo asintótico de la complejidad se ha obviado el coste computacional que tiene la función 'GetNextRule' (algoritmo 3.10). El coste de esta función es proporcional al número de ítems de un nodo (atributo 'nIndex' de la estructura '*Regla*') y con respecto al nivel en que se encuentre la regla. A medida que el nodo se va expandiendo, su valor para el atributo 'nIndex' es cada vez más pequeño, mientras que el nivel será cada vez mayor, por lo que se cumplirá que:

$$\boldsymbol{\Theta}(\text{ GetNextRule(r) }) \sim \text{MAX (r.nIndex , r.nivel)} \qquad \text{(Expresión 3.13)}$$

En cualquier caso se trata de un coste lineal que, en comparación del coste de procesar la ganancia, es despreciable y por ello no afecta al cálculo global de la complejidad del método.

UNIVERSITAS
Miguel
Hernández

Capítulo 4
Descripción de los conjuntos de datos utilizados

4.1. Visión general sobre los datos

Para probar de manera rigurosa el método propuesto en la Tesis, se han realizado dos tipos diferentes de estudios. El primero de ellos está basado en la utilización de varios conjuntos de datos simulados, con diferentes cantidades de registros (diferentes valores de 'N'), diferentes número de columnas (diferentes valores de 'C') y diferentes cantidades de posibles valores que podría tomar cada variable (diferentes valores de 'V'). El objetivo que se persigue con este primer análisis es obtener los tiempos de procesamiento del algoritmo propuesto en la Tesis bajo diferentes condiciones de carga y comprobar que, en todos los casos, dichos tiempos son coherentes con el estudio teórico de la complejidad temporal realizado en el capítulo anterior.

Para el segundo estudio se ha utilizado un conjunto de datos obtenidos a partir del registro de las transacciones de una organización empresarial real. El proveedor de dichos datos ha sido la empresa MTNG GLOBAL EXPERIENCE, S.L., más conocida por uno de sus nombres comerciales: 'MUSTANG'. Esta empresa se dedica al diseño, fabricación y venta de calzado. Con este segundo estudio se pretende confirmar la validez de los sistemas de reglas proporcionados por el método desde un punto de vista práctico para la empresa, sobre todo desde la perspectiva de que los sistemas de reglas proporcionados sean coherentes con el conocimiento del negocio y también desde el

prisma del valor estratégico añadido derivado de la incorporación de este tipo de modelos predictivos al proceso de toma de decisiones empresariales.

En este capítulo, se van a presentar los detalles de los conjuntos de datos empleados para las pruebas del algoritmo. En primer lugar se abordará el proceso de generación de los datos simulados y en segundo lugar, se comentará el origen y preprocesamiento de los datos reales provistos por la empresa MUSTANG.

4.2. Conjuntos de datos simulados

Las muestras de datos simulados se presentan como varias estructuras en forma de tabla de datos donde las filas representan los ejemplos del conjunto de entrenamiento (también llamados ítems o registros), y las columnas representan las variables o atributos que caracterizan dichos ejemplos, es decir, contienen los valores que toman las variables de los distintos ejemplos. A continuación se describe formalmente la construcción de estos conjuntos de datos, a partir de las siguientes definiciones:

- Sea el conjunto de datos 'E' que contiene 'N' ejemplos (tamaño de la muestra) y 'C' variables (columnas).
- Sea 'A_i' la variable 'i-ésima' del conjunto 'E', con i=1, 2, …, C.
- Se divide el conjunto de variables en dos subconjuntos 'E_1' y 'E_2' de tamaños 'C_1' y 'C_2', tal que $C = C_1 + C_2$.
- Sea 'a_{ij}' el valor que toma la variable 'i' en el ejemplo 'j'.
- Sea 'V' el número de los posibles valores (distintos) que puede tomar cada una de las variables 'A_i'.
- Sea '**X**' el vector que contiene todas las variables del conjunto 'E_1', tal que $\mathbf{X}=(A_1, A_2, \ldots, A_{C1})$. El número de posibles valores que puede tomar el vector '**X**' viene dado por 'V^{C1}' (se simplifica la notación definiendo 'Q' $\Leftrightarrow Q = V^{C1}$).
- Sea '$\mathbf{x_1}$', '$\mathbf{x_2}$', …, '$\mathbf{x_Q}$' los 'V^{C1}' valores que puede tomar el vector '**X**'.

Los valores para las variables 'A_1', 'A_2', …, 'A_{C1}', del conjunto 'E_1', se simulan conjuntamente. Para ello se generan 'N' instancias del vector '**X**' siguiendo los siguientes pasos:

1. Generar un vector de probabilidades '**p**', definido como $\mathbf{p} = (p1, p2, \ldots, p_Q)$ verificando que $\forall\, k \in \{1, 2, \ldots, Q\}$ se cumple que $p_k > 0$ y $\sum_k p_k = 1$
2. Generar un vector multinomial '**n**', definido como $\mathbf{n} = (n_1, n_2, \ldots, n_Q) \sim$ Multinom(N , **p**) verificando que $\sum_k n_k = N$

3. Generar sucesivamente 'n_1' valores '$\mathbf{x_1}$', 'n_2' valores '$\mathbf{x_2}$', …, 'n_Q' valores '$\mathbf{x_Q}$' y asignarlos a las filas de la muestra, de modo que se rellenen, para las 'N' filas de la muestra 'E', los datos de las 'C_1' primeras columnas.

Los valores para las variables 'A_{C1+1}', 'A_{C1+2}', …, 'A_C', del conjunto 'E_2', se simulan independientemente. $\forall i \in \{C_1+1, C_1+2, \ldots, C\}$ realizar los siguientes pasos:

1. Extraer una muestra con reemplazamiento, probabilidades iguales y tamaño 'N' del conjunto de los 'V' valores distintos que puede tomar la variable 'A_i'
2. Asignar sucesivamente los valores obtenidos a la columna 'i' de la muestra, construyéndose la columna 'i-ésima' conteniendo los valores (a_{i1}, a_{i2}, a_{iN})

Nótese que las columnas del subconjunto 'E_1' ('A_1', 'A_2', …, 'A_{C1}') se simulan utilizando una distribución de probabilidad preestablecida, '**p**', en el propio proceso de generación de valores, mientras que para simular las columnas del subconjunto 'E_2' ('A_{C1+1}', 'A_{C1+2}', …, 'A_C') se ha optado por un proceso aleatorio de generación de valores. La finalidad de este sistema es, por un lado, tener unos atributos en el conjunto de datos que sigan una cierta correlación, para luego poder verificar que los valores de soporte y confianza de las reglas generadas son coherentes con dichas probabilidades. Por otra parte, se tiene un conjunto de variables con una distribución aleatoria que fuerza a la generación de un número alto de reglas distintas para conseguir un árbol de decisión más extenso y, por tanto, computacionalmente más costoso de generar, es decir, se busca que el conjunto de reglas finales ('R'), antes de ser reducido, contenga un alto número de reglas. Así se está forzando que los datos simulados se correspondan con un caso pesimista. Por esta razón el conjunto 'E_1' de variables correlacionadas va a ser siempre pequeño, de dos columnas solamente en todos los casos ($C_1 = 2$), siendo siempre una de ellas la columna consecuente.

Finalmente, se han generado 450 conjuntos de datos para diferentes valores de 'N', 'C' y 'V' utilizando el método descrito, concretamente, para los siguientes 10 valores de 'N':

$$N \in \{2x10^5, 4x10^5, 6x10^5, 8x10^5, 10x10^5, 12x10^5, 14x10^5, 16x10^5, 18x10^5, 20x10^5\} \quad \text{(Expresión 4.1)}$$

Para 5 valores de 'C':

$$C \in \{2, 4, 8, 16, 32\} \quad \text{(Expresión 4.2)}$$

Y para 9 valores de 'V':

$$V \in \{2, 3, 4, 6, 8, 10, 12, 14, 16\} \quad \text{(Expresión 4.3)}$$

Para generar los ficheros de datos según el método descrito se desarrollaron unos sencillos scripts en lenguaje R. El algoritmo 4.1 muestra un ejemplo de uno de esos scripts para N=200000, C={2, 4, 8} y V=4.

Algoritmo 4.1: Script en R para generación de datos simulados

```
# Definición de atributos y sus posibles valores
01.  A01 <- factor(c(1,2,3,4), labels=c("a1_1","a1_2","a1_3","a1_4"))
02.  A02 <- factor(c(1,2,3,4), labels=c("a2_1","a2_2","a2_3","a2_4"))
03.  A03 <- factor(c(1,2,3,4), labels=c("a3_1","a3_2","a3_3","a3_4"))
04.  A04 <- factor(c(1,2,3,4), labels=c("a4_1","a4_2","a4_3","a4_4"))
05.  A05 <- factor(c(1,2,3,4), labels=c("a5_1","a5_2","a5_3","a5_4"))
06.  A06 <- factor(c(1,2,3,4), labels=c("a6_1","a6_2","a6_3","a6_4"))
07.  A07 <- factor(c(1,2,3,4), labels=c("a7_1","a7_2","a7_3","a7_4"))
08.  A08 <- factor(c(1,2,3,4), labels=c("CL_1","CL_2","CL_3","CL_4"))

# Generación de las columnas dependientes
09.  dat <- expand.grid(A07, A08)
10.  set.seed(1)
11.  p <- c(7344, ...)
12.  N <- 200000
13.  repet <- rmultinom(1, N, p)
14.  cumrep <- cumsum(repet)
15.  cumrep <- c(0,cumrep)
16.  E <- dat[1,]
17.  k <- 1
18.  for(j in 1:nrow(dat)) {
19.      print(repet[j])
20.      if(repet[j]!=0) {
21.          for(i in 1:repet[j]) {
22.              E[k,] <- dat[j,]
23.              k <- k+1
24.          }
25.      }
26.  }

# Generación de las columnas independientes
27.  A_01 <- sample(A01, N, replace = TRUE)
28.  A_02 <- sample(A02, N, replace = TRUE)
29.  A_03 <- sample(A03, N, replace = TRUE)
30.  A_04 <- sample(A04, N, replace = TRUE)
31.  A_05 <- sample(A05, N, replace = TRUE)
32.  A_06 <- sample(A06, N, replace = TRUE)

# Generación de los ficheros de datos
33.  write.table(E,file="E01.txt",row.names=FALSE,sep="\t",quote=FALSE)
34.  E <- cbind(A_06, E)
35.  E <- cbind(A_05, E)
36.  write.table(E,file="E02.txt",row.names=FALSE,sep="\t",quote=FALSE)
37.  E <- cbind(A_04, E)
38.  E <- cbind(A_03, E)
39.  E <- cbind(A_02, E)
40.  E <- cbind(A_01, E)
41.  write.table(E,file="E03.txt",row.names=FALSE,sep="\t",quote=FALSE)
42.  rm(list=ls(all=TRUE))
```

Según se desprende del algoritmo de ejemplo 4.1, se generan tres ficheros de datos, 'E01.txt', 'E02.txt' y 'E03.txt' de 2, 4 y 8 columnas respectivamente. Esas columnas

(‘A01’, ‘A02’, …, ‘A08’) se definen al principio del código, todas con 4 posibles valores, identificando las etiquetas de dichos valores. La última de esas columnas se usará como variable de clase o consecuente (valores ‘CL_1’, ‘CL_2’, ‘CL_3’ y ‘CL_4’). Con dichas columnas se genera una matriz de datos únicamente con las dos últimas a las que se van asignando valores según una determinada distribución de probabilidad ‘**p**’. Posteriormente se generan los valores aleatorios de los seis primeros atributos. El primero de los ficheros de datos (‘E01.txt’) se generan a partir de la matriz de datos original, únicamente con 2 columnas; se añaden a dicha matriz los atributos ‘A06’ y ‘A05’ y se genera el segundo fichero de datos (‘E02.txt’). Por último, se añaden a la matriz de datos las columnas restantes, ‘A04’, ‘A03’, ‘A02’, y ‘A01’ y se genera el último fichero (‘E03.txt’) de 8 columnas.

4.3. Datos de venta B2B

Los datos proporcionados por la empresa MUSTANG son los correspondientes a los pedidos realizados por sus clientes minoristas (consumidores no finales). Es decir se trata de datos de ‘*retail*’ o B2B (‘*Business-to-Business*’). Se dispone de todos los pedidos recibidos desde octubre de 2011 hasta junio de 2013 para los diversos productos comercializados, tanto nacionales como internacionales, que en total suman 423085 registros.

El objetivo, desde el punto de vista empresarial, es poder predecir el número de unidades pedidas de los diversos productos en función de de las características de los mismos. En la tabla 4.1 se muestra una descripción de los campos o variables del conjunto de datos original suministrado por la empresa, antes de determinar cuáles de ellos debían ser considerados para el análisis y cuales obviados. A continuación se indica cuáles, de entre las variables disponibles fueron descartadas tras consensuarlo con los expertos de la empresa.

El algoritmo CREA trabaja con variables discretas o discretizadas para generar conjuntos de reglas de clasificación (ver más adelante el tratamiento realizado con los campos ‘*Fecha pedido*’, ‘*Unidades*’ y ‘*Precio par*’). Los modelos basados en este tipo de reglas son de difícil interpretación cuando las variables utilizadas tienen un elevado número de valores (varios centenares o miles) con una alta dispersión, o que no pueden ser agrupados mediante un criterio claro. También ocurre que entre dichas variables hay algunas (como ‘*Pedido*’ o ‘*ID línea*’) que no aportan información importante desde el punto de vista de la toma de decisiones, por lo que se optó por eliminarlas del estudio (sombreadas en rojo en la tabla 4.1). El campo ‘*Familia artículo*’ contiene el valor combinado de ‘*Marca*’, ‘*Temporada*’ y ‘*Estrategia*’, por lo que se decide eliminarlo también del estudio ya que aportaría información redundante (sombreado en naranja en la tabla 4.1).

Tabla 4.1: Variables disponibles en el conjunto de datos de MUSTANG

Campo	Nº valores	Comentario
Empresa	4	
Pedido	57431	
Tipo Documento	9	
Fecha Pedido	554	Valor de tipo **DATE** (casi un valor diario)
Agentes	64	
Clientes	2945	
Código artículo	40456	
Modelo artículo	2508	
Horma artículo	562	
Estilo artículo	4	
Tipo artículo	14	
Corte artículo	25	
Rasgo artículo	29	
Marca artículo	7	
Familia artículo	219	Combinación de “Marca” + “Temporada” + “Estrategia”
Categoría artic.	3	
Color	3035	Hay códigos diferentes que se refieren al mismo color
Unidades	303	Valor de tipo **INTEGER**
Precio par	593	Valor de tipo **REAL**
Divisas	2	
Cambio	4	Valor de tipo REAL
Departamento	12	
Temporada	14	Combinación de “año” + {1,2} (donde {1,2} = campaña)
Estrategias	42	
Secciones	4	
Línea artículo	6	
País	50	
Provincia	89	Para pedidos en España
ID línea	~420000	Pseudo-identificador de la tabla

Otros atributos como '*Empresa*', '*Tipo Documento*', '*Departamento*', '*Estrategia*', '*Sección*' o '*Línea artículo*' (marcados en amarillo en la tabla 4.1), se caracterizan porque pudiendo tomar varios valores, hay uno de ellos que predomina en exceso sobre los demás, por lo que incluirlas producirá un sesgo en los resultados, así que también se opta por eliminarlas del estudio. En la tabla 4.2 se muestra un ejemplo de dichos valores.

Tabla 4.2: Ejemplo de variables con un valor predominante

Variable 'Empresa'

Empresa	Nº valores
EUROPE	421356
GLOBAL LTD	1179
PASCO	326
USA	224

Variable 'Tipo Documento'

Tipo Doc	Nº valores
PINA	324585
PEDI	26964
PEXD	23585
PEXS	19146
PVEN	15587
PEDR	12273
PEXP	692
PUSA	224
PEPC	29

Las variables '*Estilo*', '*Marca*', '*Categoría*' y '*Rasgo*' presentan una problemática idéntica a la descrita, pero por ser consideradas por la empresa cliente como especialmente relevantes para determinar características del producto, se decide explícitamente no descartarlas en el estudio.

Existen otras variables '*Agente*', '*Divisa*', '*Cambio*', '*Pais*' y '*Provincia*' (marcadas en azul en la tabla 4.1) que también se descartan a criterio del proveedor por diversos motivos. En general se considera que es información no relevante para caracterizar los productos, por lo tanto se pueden descartar del análisis.

Adicionalmente, existen tres variables '*Fecha pedido*', '*Precio par*' y '*Unidades*' de naturaleza numérica (o pseudo-numérica en el caso de '*Fecha pedido*') que se deben discretizar. En el caso de la fecha, a sugerencia del proveedor de los datos, se considera segmentar los valores de dicha variable por semanas, como se puede observar en la tabla 4.3 se dispone de un valor por cada semana (sin considerar el año), se optó por este método de discretización ya que el año está incluido en la información proporcionada por el campo '*Temporada*'. Para la empresa hay dos campañas de ventas al año etiquetadas con los valores '1' y '2', ese valor junto con los dos últimos dígitos del año forman los valores de la variable '*Temporada*', de tal forma que, por ejemplo, los valores '121' y '122' se refieren respectivamente a la primera y segunda campaña del año 2012.

Tabla 4.3: Segmentación del campo 'Fecha pedido' a 'Semana Ped'

Sem-Ped	Nº valores	Sem-Ped	Nº valores
Sem_01	1471	Sem_27	3339
Sem_02	2906	Sem_28	3979
Sem_03	4405	Sem_29	2030
Sem_04	5040	Sem_30	1952
Sem_05	5146	Sem_31	1814
Sem_06	3254	Sem_32	937
Sem_07	3355	Sem_33	609
Sem_08	4266	Sem_34	1682
Sem_09	4831	Sem_35	2170
Sem_10	8099	Sem_36	2752
Sem_11	11142	Sem_37	3222
Sem_12	7028	Sem_38	5115
Sem_13	12588	Sem_39	4893
Sem_14	15388	Sem_40	11456
Sem_15	16647	Sem_41	12491
Sem_16	19340	Sem_42	18219
Sem_17	17222	Sem_43	16289
Sem_18	11425	Sem_44	12966
Sem_19	15936	Sem_45	17737
Sem_20	22654	Sem_46	12509
Sem_21	15613	Sem_47	12677
Sem_22	11414	Sem_48	9576
Sem_23	9505	Sem_49	4805
Sem_24	6857	Sem_50	6755
Sem_25	7264	Sem_51	2953
Sem_26	4613	Sem_52	2749

Finalmente, el campo '*Precio par*' se discretiza en los siguientes 5 rangos de valores: [0-25], [26-50], [51-75], [76-100] y [>100]; y por otra parte, el campo '*Unidades*' se segmenta en otros 6 rangos: [0-15], [16-30], [31-50], [51-100], [101-200] y [>200]. Para disponer de una descripción más detallada, en las tablas 4.4 a 4.12, se muestra como quedan todas las variables que finalmente se van a considerar en el estudio (excepto la variable '*SemPed*' vista en la tabla 4.3).

Tabla 4.4: Variable 'Estilo artículo'

Estilo	Nº valores
est_02	343479
est_01	70134
est_	9434
est_10	38

Tabla 4.5: Variable 'Tipo artículo'

Tipo	Nº valores
tip_07	89295
tip_02	74291
tip_01	69558
tip_03	58371
tip_09	51908
tip_06	31510
tip_04	24426
tip_05	11497
tip_	9434
tip_10	1915
tip_08	366
tip_15	324
tip_13	152
tip_73	38

Tabla 4.6: Variable 'Corte artículo'

Corte	Nº valores
cor_12	89295
cor_10	74176
cor_05	56739
cor_03	49028
cor_14	31628
cor_02	28682
cor_11	20443
cor_07	19460
cor_21	11614
cor_09	11373
cor_	9434
cor_23	6102
cor_06	4966

Corte	Nº valores
cor_01	2915
cor_22	2301
cor_15	1915
cor_04	1747
cor_25	324
cor_18	259
cor_26	242
cor_17	152
cor_08	124
cor_13	107
cor_0	38
cor_20	21

Tabla 4.7: Variable 'Rasgo artículo'

Rasgo	Nº valores
ras_01	203545
ras_03	35905
ras_06	27941
ras_09	25772
ras_07	21120
ras_11	16957
ras_04	16541
ras_12	14622
ras_08	13862
ras_10	10383
ras_	9434
ras_02	9107
ras_15	7465
ras_05	5366
ras_48	3683

Rasgo	Nº valores
ras_20	899
ras_46	149
ras_25	92
ras_14	51
ras_28	38
ras_23	37
ras_33	34
ras_18	28
ras_39	17
ras_35	13
ras_32	9
ras_19	8
ras_82	6
ras_89	1

Tabla 4.8: Variable 'Marca artículo'

Marca	Nº valores
MT	342072
MM	48837
SX	21777
EM	5495
MB	4190
CH	705
MV	9

Tabla 4.9: Variable 'Categoría artículo'

Categoría	Nº valores
Z	368574
C	54470
P	41

Tabla 4.10: Segmentación del campo 'Unidades' a 'UnidadesD'

UnidadesD	Nº valores
Num_000_015	69897
Num_016_030	85907
Num_031_050	205961
Num_051_100	30592
Num_101_200	19915
Num_200_+++	10813

Tabla 4.11: Segmentación del campo 'Precio par' a 'PrecioParD'

PrecioParD	Nº valores
pre_000_025	74275
pre_025_050	188852
pre_050_075	123454
pre_075_100	14545
pre_100_+++	21959

Tabla 4.12: Variable 'Temporada'

Temporada	Nº valores
tem_122	120453
tem_131	120194
tem_121	103537
tem_132	69473
tem_112	5582
tem_111	2531
tem_102	649
tem_101	302
tem_092	144
tem_000	108
tem_091	81
tem_071	19
tem_082	7
tem_081	5

Sobre los datos expuestos en las tablas anteriores cabe resaltar lo siguiente: Para la variable discretizada '*UnidadesD*' (tabla 4.10) se tiene que uno de sus valores, 'Num_031_050', se repite más de doscientas mil veces (casi el 50% de los casos), esto introduce un sesgo en los datos que afectará a los resultados del análisis. Normalmente, una situación como esta puede resolverse escogiendo un criterio de discretización más adecuado. En la práctica, para este caso concreto no ha sido posible ya que en los datos originales, antes de la discretización, se tiene que el valor de unidades vendidas '36' se repite 196.798 veces, es decir, los datos no agregados ya incorporan un sesgo imposible de salvar independientemente del criterio de discretización escogido.

Por otra parte, en la variable '*Temporada*' se observa que hay valores como 'tem_000', 'tem_07X', 'tem_08X', 'tem_09X' y 'tem_10X'. Estos valores deben interpretarse como correspondientes a pedidos realizados en los años 2000, 2007, 2008, 2009 y 2010 (donde 'X' puede ser cualquiera de los valores '1' o '2' correspondientes a las campañas del año). Esto se contradice con el hecho de que los datos proporcionados se corresponden a pedidos realizados entre 2011 y 2013. Entiéndase que dichos valores son anómalos, es decir, al menos en la variable '*Temporada*' se tiene que hay ruido en los datos.

A modo de resumen, en la tabla 4.13 se muestra una descripción del conjunto de datos que finalmente va a utilizarse para el análisis con datos reales.

Tabla 4.13: Descripción del conjunto de datos empresariales

Número de registros: N				N = 423085
Número de columnas: C				C = 10
Número de valores por columna: V				
	V1	=	|Sem-Ped|	= 52
	V2	=	|Marca|	= 7
	V3	=	|PrecioParD|	= 5
	V4	=	|Temporada|	= 14
	V5	=	|Categoría|	= 3
	V6	=	|Estilo|	= 4
	V7	=	|Tipo|	= 14
	V8	=	|Corte|	= 25
	V9	=	|Rasgo|	= 29
	V10	=	|UnidadesD|	= 6

Capítulo 5
Experiencia computacional

5.1. Visión general de los experimentos

Para comprobar la eficacia y eficiencia del algoritmo CREA se han realizado varios experimentos con múltiples conjuntos de datos. Estos han permitido probar el método exhaustivamente desde dos perspectivas diferentes:

- Complejidad computacional del algoritmo: Realización de varias ejecuciones del proceso bajo diferentes condiciones de carga, anotando la duración de cada ejecución y verificando que los tiempos observados son coherentes con el estudio teórico de la complejidad computacional realizado en el capítulo 3. Es decir, se comprobará que los tiempos son compatibles con la expresión 3.12.

$$\boldsymbol{\Theta}(\text{CREA}) = \begin{cases} C^2 \text{ x } V^2 \text{ x } N & \Leftrightarrow \ N < V^C \\ V^{C-1} & \Leftrightarrow \ V^C \leq N \leq V^{C+2} \\ C^2 \text{ x } N & \Leftrightarrow \ N > V^{C+2} \end{cases} \qquad \text{(Repetición expresión 3.12)}$$

- Coherencia de los sistemas de reglas proporcionados: Se verifica que las reglas obtenidas reflejan la realidad de los datos de entrada suministrados. Con ello se

debe comprobar si el modelo generado contiene todas las reglas realmente importantes y si ha descartado las menos relevantes, y solamente estas.

El algoritmo CREA ha sido implementado en el lenguaje C++ y los tiempos de ejecución se han calculado haciendo uso de la función 'clock()' de la librería estándar de C/C++ '<time.h>' que permite obtener los '*clocks*' (ciclos) del reloj interno del procesador para el programa que se está ejecutando. Esta función da una medida en milisegundos del tiempo que ha transcurrido desde que un programa se está ejecutando. Así, para saber la duración de un determinado proceso, basta con tomar nota del número de 'clocks' con la función 'clock()' antes y después de la ejecución del proceso que se quiere medir y restar ambos tiempos. Adicionalmente, en un ordenador, además del propio proceso objeto de estudio, se están ejecutando simultáneamente otras tareas del sistema operativo, dichas tareas introducen un pequeño sesgo (por otra parte inevitable), que puede hacer variar el tiempo de dos ejecuciones de un mismo proceso, incluso bajo las mismas condiciones de carga. Para mitigar el error producido por este sesgo, en todos los procesos de medición realizados se ha ejecutado el mismo experimento, con los mismos datos de entrada, hasta 10 veces para calcular un tiempo promedio. El algoritmo 5.1 muestra un ejemplo de cómo sería el código en C/C++ para realizar este cálculo.

Algoritmo 5.1: Cálculo de tiempos en C/C++

```
 1.  #include<stdio.h>
 2.  #include<time.h>
 3.
 4.  main()
 5.  {
 6.    FILE *f = fopen("ficheroTiempos.txt", "w");
 7.    clock_t ini, fin;
 8.    int i;
 9.
10.    for(i = 0 ; i<10 ; i++)
11.    {
12.      ini = clock();
13.      PROCESO();     //función cuyo tiempo se quiere calcular
14.      fin = clock();
15.      fprintf(f, "%d\n", (fin-ini));
16.    }
17.
18.    fclose(f);
19.  }
```

Este algoritmo 5.1 calcula el tiempo de ejecución de la función '*PROCESO( )*' (línea 13). Como puede verse, dicho cálculo se realiza 10 veces (bucle '*for*', línea 10) y los tiempos obtenidos se van guardando en un fichero de texto de nombre '*ficheroTiempos.txt*' (creado en la línea 6 y actualizado en la línea 15). Posteriormente, dicho fichero de tiempos puede ser importado a una hoja de cálculo para hacer las operaciones que se estimen oportunas.

Para realizar el primero de los estudios propuestos, el cálculo empírico de la complejidad computacional, se van a utilizar los 450 conjuntos de datos simulados descritos en el epígrafe 4.2. El objetivo es analizar cómo evolucionan los tiempos de ejecución para diferentes valores de 'N', 'C' y 'V'. Para probar la validez de los sistemas de reglas proporcionados por el algoritmo CREA se recurre a un conjunto de datos reales de ventas de la empresa del sector del calzado MUSTANG (descritos en el apartado 4.3). El resultado obtenido se ha sometiendo a juicio de expertos en el ámbito empresarial y del negocio, siendo contrastado por ejecutivos y responsables del área de sistemas de la propia compañía que ha cedido los datos. Estos confirmaron la validez y coherencia de los modelos proporcionados por el método CREA. Para este conjunto de datos también se ha comprobado que los tiempos de cómputo son coherentes con el caso de los datos simulados y también con la complejidad teórica calculada en la sección 3.3.

5.2. Resultados de CREA sobre datos simulados

Como ha quedado patente en el estudio teórico de la complejidad temporal del algoritmo CREA, dado un conjunto de ejemplos 'E' (datos de entrenamiento para generar un modelo de reglas de clasificación) existen tres parámetros que caracterizan dicho conjunto que van a influir en el tiempo de ejecución del proceso de generación de reglas; estos parámetros son el número de ejemplos 'N' (filas/registros del conjunto de datos 'E'), el número de atributos 'C' de cada ejemplo (columnas del conjunto de datos) y el número de valores 'V' que puede tomar cada atributo (asumiendo que todas las columnas toman siempre un número idéntico de valores). Por tanto, a partir de los tiempos de procesamientos medidos empíricamente se puede construir un cubo de datos en tres dimensiones. Estas serían los atributos 'N', 'C' y 'V', y los diversos valores de tiempo observados ocuparían las diferentes posiciones del citado cubo (ver figura 5.1). En adelante se llamará cubo [NCV-T] al mencionado cubo de datos.

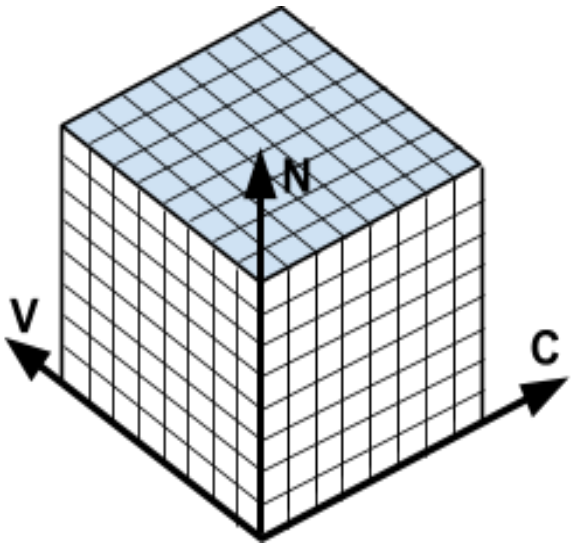


Figura 5.1: Representación gráfica del cubo de datos [NCV-T] de tres dimensiones

Ese enfoque permite plantear, de forma muy intuitiva, un análisis multidimensional del tiempo de procesamiento, que es la variable objeto de estudio. En el Anexo 1 de esta Tesis se encuentran todas las gráficas obtenidas en el exhaustivo estudio que se ha realizado. En este epígrafe se presenta un extracto del citado anexo con las conclusiones más significativas. Nótese que el cubo [NCV-T] puede ser girado y desplegado de múltiples formas y ser presentado en dos dimensiones con apariencias distintas. Se define como 'vista' a cada posible modo en que el cubo [NCV-T] puede ser presentado en dos dimensiones. La tabla 5.1 presenta una posible vista de este cubo con los tiempos de procesamiento medidos para los 450 conjuntos de datos considerados. Se observa que la variable 'N' se plasma en las 10 columnas de datos de la tabla, y los parámetros 'C' y 'V' en las filas, las cuales están agrupadas por los 5 posibles valores de 'C', por lo que los datos presentados en esta vista aparecen estructurados en 5 bloques de 9 filas, una por cada valor de 'V'.

Tabla 5.1: Vista 2D del cubo [NCV-T] con los tiempos de ejecución de CREA

C	V	N = $2x10^5$	N = $4x10^5$	N = $6x10^5$	N = $8x10^5$	N = $10x10^5$	N = $12x10^5$	N = $14x10^5$	N = $16x10^5$	N = $18x10^5$	N = $20x10^5$
2	**2**	21.9	42.2	62.5	82.8	104.7	123.4	145.3	165.7	185.9	206.2
2	**3**	26.5	51.5	78.1	103.1	129.6	154.6	179.7	207.8	232.8	257.8
2	**4**	31.2	60.9	92.2	123.5	154.7	184.4	217.2	246.9	278.1	307.8
2	**6**	40.6	81.2	121.8	162.5	203.2	245.3	286.0	328.1	368.7	410.9
2	**8**	50.0	101.6	153.1	201.5	253.1	303.1	354.7	406.2	457.8	507.8
2	**10**	60.9	121.9	181.2	242.2	301.6	362.5	423.4	486.0	545.3	607.8
2	**12**	71.9	143.7	214.0	286.0	357.8	426.6	500.0	571.9	642.2	717.2
2	**14**	82.8	165.6	246.9	329.7	411.0	493.8	576.6	657.8	740.7	826.5
2	**16**	93.7	185.9	279.7	371.9	464.1	556.3	648.4	742.2	835.9	929.6
4	**2**	85.9	173.4	259.4	348.4	435.9	521.9	609.4	693.7	782.8	870.3
4	**3**	106.3	215.6	325.0	434.4	546.9	654.6	768.7	876.5	987.5	1100.0
4	**4**	123.4	251.5	384.3	510.9	671.9	778.1	940.7	1082.8	1203.1	1334.3
4	**6**	160.9	314.1	470.3	626.5	865.6	951.6	1193.8	1446.9	1523.5	1864.1
4	**8**	206.3	390.7	581.3	768.8	967.2	1145.3	1345.3	1584.4	1739.1	2017.2
4	**10**	286.0	518.8	751.6	984.3	1228.2	1473.4	1720.3	1984.4	2209.4	2504.7
4	**12**	409.4	675.0	957.8	1214.1	1485.9	1743.8	2004.7	2284.3	2551.5	2895.3
4	**14**	579.6	892.1	1215.6	1529.7	1836.0	2121.8	2400.0	2678.1	2984.4	3853.1
4	**16**	873.5	1292.2	1623.4	1971.9	2348.4	2704.7	3029.7	3385.9	3740.6	4081.3
8	**2**	356.2	742.2	1143.8	1454.7	1803.1	2175.0	2559.4	2921.8	3400.0	3693.7
8	**3**	493.7	1079.7	1589.1	2121.9	2715.6	3190.6	3779.7	4337.5	4887.5	5475.0
8	**4**	1139.1	1871.9	2548.4	3243.8	3995.3	4635.9	5386.0	6084.4	6714.0	7543.8
8	**6**	2592.2	4846.9	6782.8	9262.5	12661.0	15840.7	18390.7	15682.8	16901.6	18626.6
8	**8**	3039.1	5911.0	8765.6	11881.3	14517.2	16981.2	19621.9	22870.3	25232.9	27082.8
8	**10**	3623.4	7178.1	9773.4	14190.6	16042.2	20476.5	24728.1	28907.8	31031.3	34503.1
8	**12**	3543.8	7451.6	10486.0	15240.7	16406.2	21496.9	26257.8	30632.8	28815.7	33482.8
8	**14**	3592.2	7512.5	10937.5	15931.3	17378.1	22814.1	27834.4	33500.0	30514.1	35579.6
8	**16**	3746.9	7729.7	11017.2	16425.0	18003.1	23561.0	29109.4	35428.2	32306.3	37560.9
16	**2**	2961.0	5690.6	7904.7	10278.1	12220.3	14364.1	16687.5	19207.8	21746.8	24231.3
16	**3**	4564.0	10354.6	14387.5	24481.2	29262.5	33265.6	45817.2	59315.6	75893.7	64739.1
16	**4**	4231.3	9300.0	17284.4	20937.5	29150.0	39946.9	52575.0	46112.5	51581.3	62971.9
16	**6**	4390.6	9157.8	14093.8	19545.3	26634.4	35192.2	36668.8	43014.1	46409.3	55281.2
16	**8**	4382.8	9521.8	15398.5	20534.3	25409.4	30795.4	34679.7	40021.9	45426.6	51703.1
16	**10**	5456.2	10726.6	16198.4	22057.8	28173.5	35743.7	42332.9	49861.0	56236.0	62960.9
16	**12**	5435.9	12235.9	18570.3	24479.7	30231.3	36056.3	40965.7	47335.9	54375.0	60689.1
16	**14**	5989.1	11914.1	19436.0	27309.4	34656.3	42089.1	48403.1	55364.1	62090.7	68067.2
16	**16**	7239.1	12782.8	19184.3	26917.2	34836.0	43226.5	51579.7	60587.5	69168.8	78279.7
32	**2**	10381.2	24142.2	34725.0	54681.2	59109.4	80293.8	103779.7	130282.8	129020.3	136096.9
32	**3**	9246.9	21984.4	31412.5	51523.4	54103.1	72500.0	95192.2	120854.7	103211.0	120107.8
32	**4**	9296.9	20646.9	28576.5	47801.6	69915.7	63828.1	83643.7	106926.6	130068.7	156540.6
32	**6**	9432.8	19364.1	29368.7	45698.5	51856.2	68189.1	87121.8	105979.7	100217.2	112842.2
32	**8**	8443.7	18787.5	30290.6	40454.7	50653.1	60165.6	70648.4	80859.4	89837.5	100078.1
32	**10**	10832.8	21506.3	31565.6	43417.2	56071.9	70656.3	85953.2	101003.1	114484.4	128120.3
32	**12**	10207.8	24414.0	38435.9	49787.5	61879.7	72765.6	84560.9	92729.7	105367.2	118275.0
32	**14**	11490.6	22843.8	38571.9	55395.3	72539.1	88464.0	103993.8	116043.7	129711.0	142379.7
32	**16**	14615.6	25098.4	37075.0	52078.2	69934.3	88725.0	108309.4	126728.1	146712.5	166485.9

En adelante, se denominará ‘vista 1’ a la vista de la tabla 5.1. Cambiando el orden de las filas de la ‘vista 1’ se puede conseguir otra vista del cubo [NCV-T]. Si las filas se agrupan por los valores de ‘V’ se obtiene una distribución de los datos agrupados en 9 bloques (valores de ‘V’) de 5 filas cada bloque (valores de ‘C’). En adelante, a esta segunda vista así construida, se la denominará ‘vista 2’ (la tabla AI.2 del Anexo I muestra cómo se verían los datos de la ‘vista 2’). Esquemáticamente, se puede representar ambas vistas como se indica en la figura 5.2.

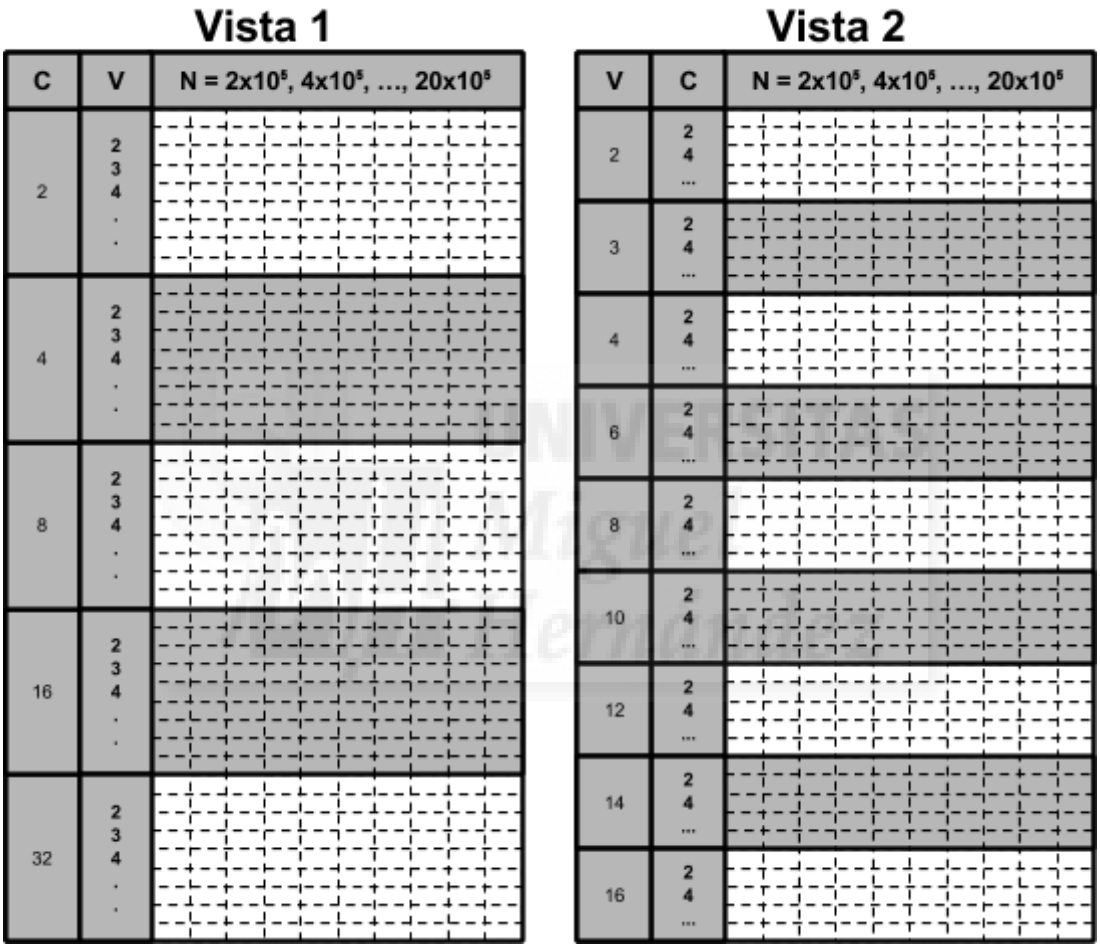


Figura 5.2: Representación esquemática de la ‘vista 1’ y la ‘vista 2’ del cubo [NCV-T]

El planteamiento multidimensional del análisis realizado requiere hacer tres tipos de estudios, uno por cada dimensión del cubo [NCV-T]. En cada uno de ellos se fijarán dos de las dimensiones a un determinado valor y se hará variar la tercera para estudiar cómo evoluciona el tiempo de procesamiento. Por tanto estos estudios son:

- Estudio 1: Análisis del tiempo de ejecución en función de ‘N’ (fijando ‘C’ y ‘V’).
- Estudio 2: Análisis del tiempo de ejecución en función de ‘C’ (fijando ‘N’ y ‘V’).
- Estudio 3: Análisis del tiempo de ejecución en función de ‘V’ (fijando ‘N’ y ‘C’).

Adicionalmente, cada estudio se ha realizado en dos partes, por las dos variables que quedan fijadas. En cada una se presentan resultados con gráficos en los que una de las variables se fija a un valor concreto para dicha gráfica y la otra se usa como identificador de las diversas series de datos representadas que permiten un análisis comparativo. Las vistas 1 y 2 descritas anteriormente van a servir para indicar visualmente cuál es el origen de datos de cada una de las gráficas que se muestran en el análisis, en adelante "Gráfica 1", "Gráfica 2", …, "Gráfica x", que se verán en varias de las figuras siguientes, hacen referencia a gráficas que se pueden encontrar en el Anexo I. En este capítulo se muestran solamente aquellas gráficas que se han considerado más significativas para la comprensión de la experiencia computacional.

5.2.1 Estudio 1: Análisis del tiempo frente a 'N' Parte 1: Comparativa por el número de columnas 'C'

La figura 5.3 muestra el origen de las series de datos del análisis realizado para esta parte. Se han generado gráficas en las que se puede comparar los tiempos de ejecución para los diversos valores de 'V'.

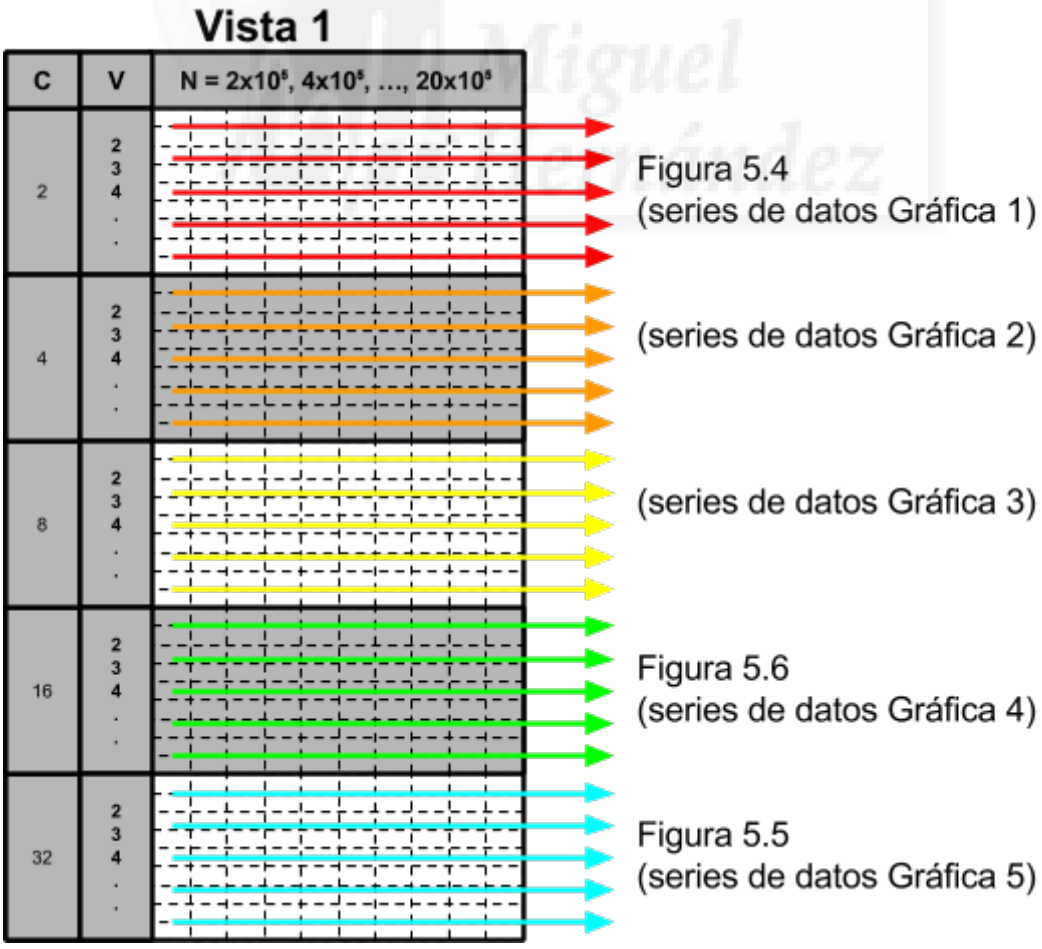


Figura 5.3: Origen de las series de datos para el "Estudio 1 - Parte 1"

Las figuras 5.4 y 5.5 ilustran cómo al fijar un valor de ‘C’ los tiempos de procesamiento del algoritmo CREA siguen una tendencia lineal a medida que aumenta el número de registros del conjunto de datos ‘N’.

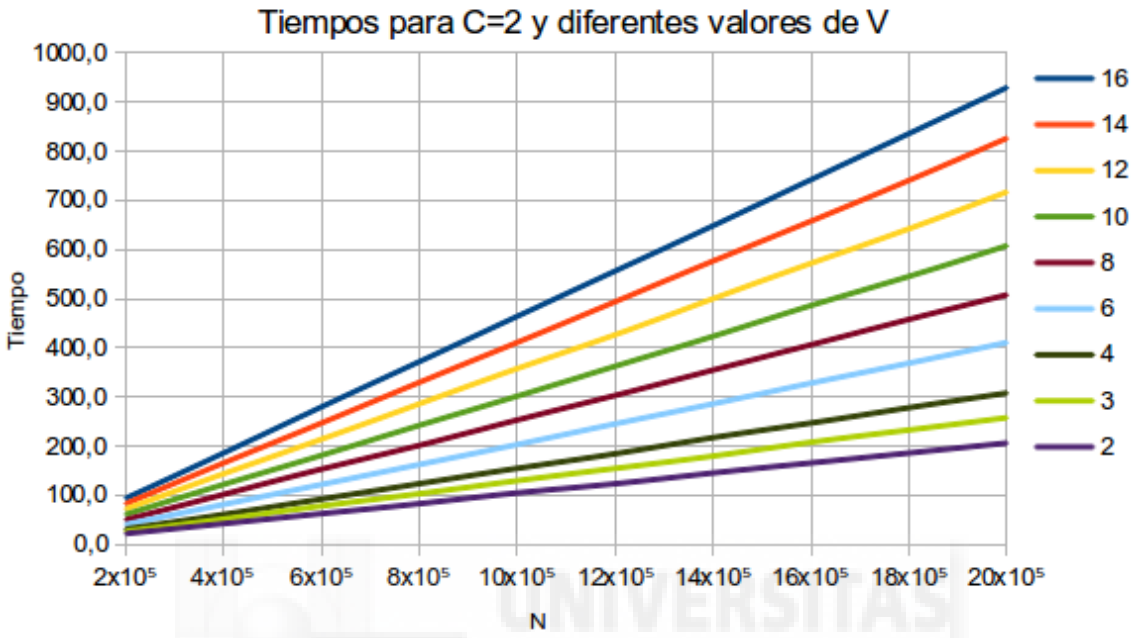


Figura 5.4: Tiempo frente a ‘N’ para ‘C=2’

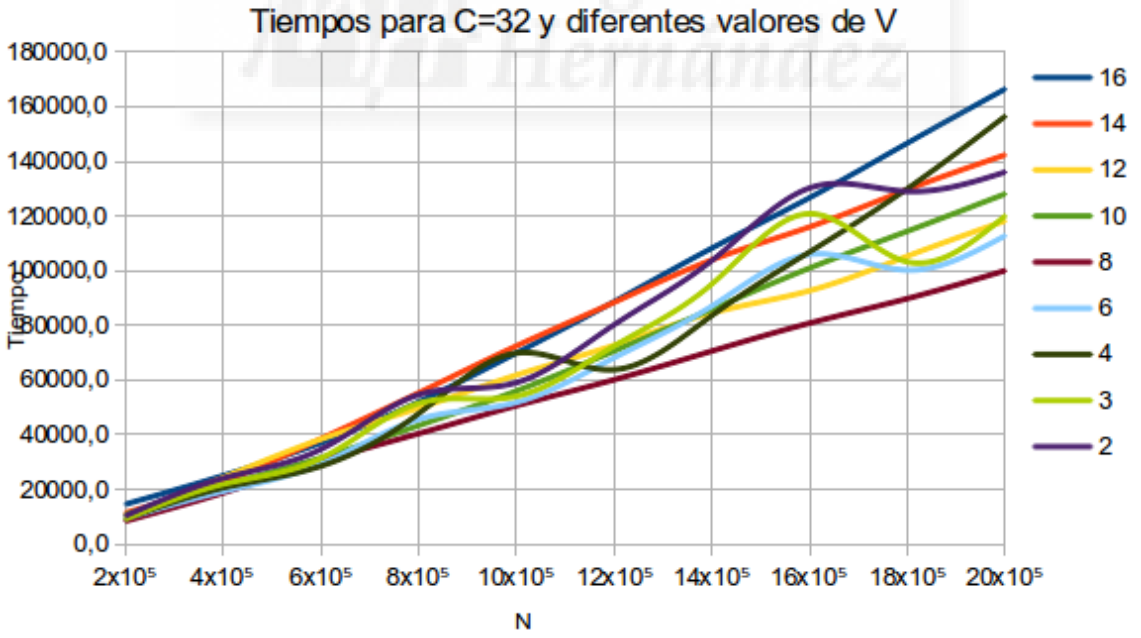


Figura 5.5: Tiempo frente a ‘N’ para ‘C=32’

En estas gráficas se ve como los tiempos de ejecución son mucho mayores al aumentar el número de columnas ‘C’ del conjunto de datos. Mientras que en la figura 5.4 (‘C=2’) ningún tiempo sobrepasa los 1000 clocks de reloj, en la figura 5.5 (‘C=32’), para el valor de ‘N’ más pequeño los tiempos están todos próximos o por encima de 10000

clocks. También se puede observar como en la primera gráfica la tendencia lineal es muy estable, mientras que en la segunda se ven algunas fluctuaciones, pero la tendencia general sigue siendo lineal. También se aprecia que, mientras en la primera gráfica, 'C=2', un aumento en el número de valores siempre implica un aumento del tiempo, en la segunda gráfica, con un mayor número de columnas, los tiempos experimentan ciertas fluctuaciones atribuibles a una mayor dependencia de la naturaleza del conjunto de datos analizado con respecto a los parámetros 'N', 'V' o 'C'.

En otras gráficas obtenidas en esta parte del análisis, como por ejemplo la que se muestra en la figura 5.6 ('C=16'), se observa una diferencia significativa de la pendiente de los tiempos de ejecución de algunos casos con respecto a otros.

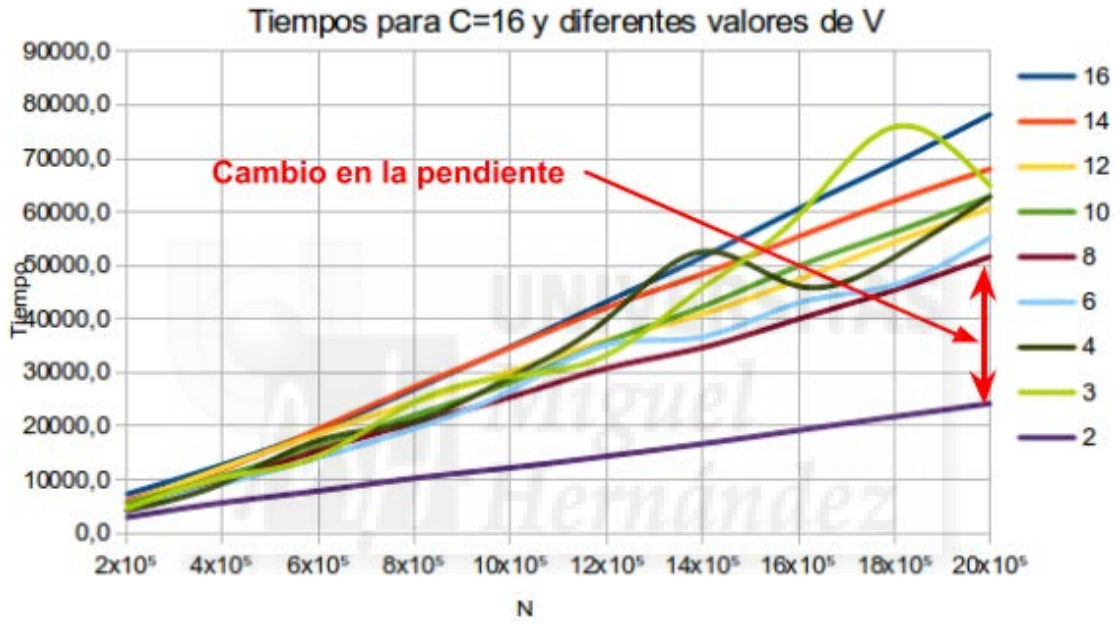


Figura 5.6: Tiempo frente a 'N' para 'C=16'

Para 'V=2' la pendiente observada es notoriamente menor que para el resto de casos. Nótese que para conjuntos de datos con 2 valores se cumple que:

$$V^{C+2} = 2^{18} = 262144 \qquad \text{(Expresión 5.1)}$$

Este es un valor menor que todos los valores de 'N' considerados (excepto el primero), por lo que para el caso de conjuntos de datos con 2 valores se estaría en el tercer tramo de la función de complejidad (expresión 3.12), es decir, pendiente 'C^2', mientras que para 'V=3' se tiene que:

$$V^{C} = 3^{16} = 43046721 \qquad \text{(Expresión 5.2)}$$

Este valor es mucho mayor que todos los valores de 'N' considerados. En este caso se estaría en el primer tramo de la función de complejidad donde la pendiente esperada para la curva de tiempos es 'C^2xV^2', mayor que en el caso de 'V=2'.

5.2.2 Estudio 1: Análisis del tiempo frente a ‘N’ Parte 2: Comparativa por el número de valores ‘V’

La figura 5.7 indica cuál ha sido la elección de las series de datos del análisis realizado en esta segunda parte del primer estudio. Se han generado gráficas en las que se ha fijado un valor de ‘V’ y es posible comparar los tiempos de ejecución del algoritmo CREA para los diversos valores de ‘C’.

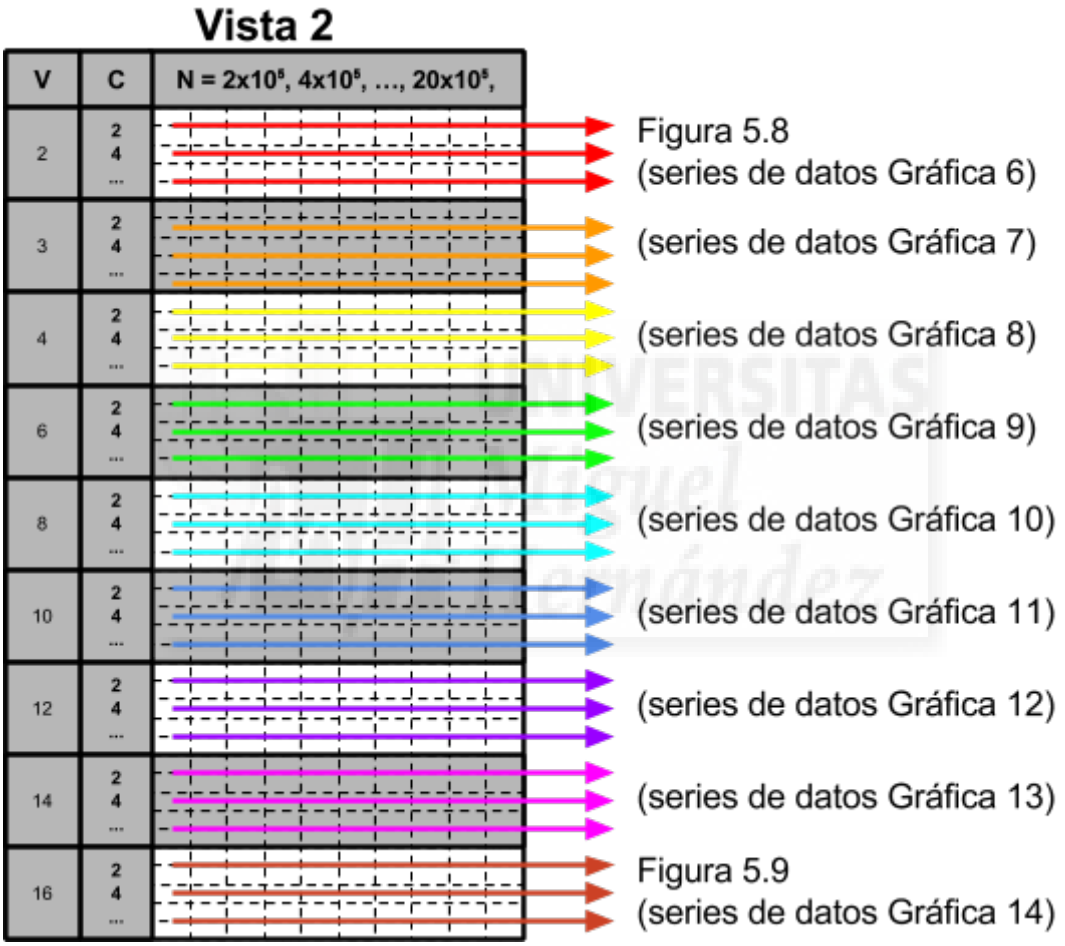


Figura 5.7: Origen de las series de datos para el “Estudio 1 - Parte 2”

Las gráficas obtenidas en esta parte del estudio muestran de forma más notoria como cambia la pendiente de las curvas representadas para los diferentes valores de ‘C’. Se verifica como al aumentar el número de columnas también lo hace la pendiente de las líneas de tiempos. En todos los casos el comportamiento observado es lineal con respecto a ‘N’ (con la salvedad de las fluctuaciones mencionadas en el epígrafe anterior).

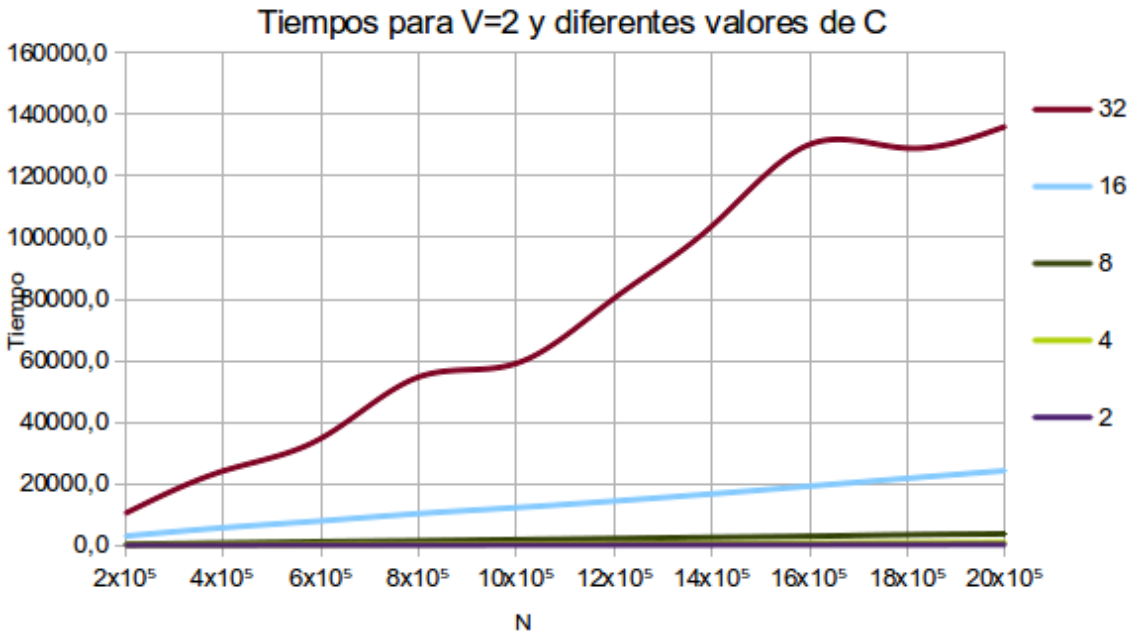


Figura 5.8: Tiempo frente a 'N' para 'V=2'

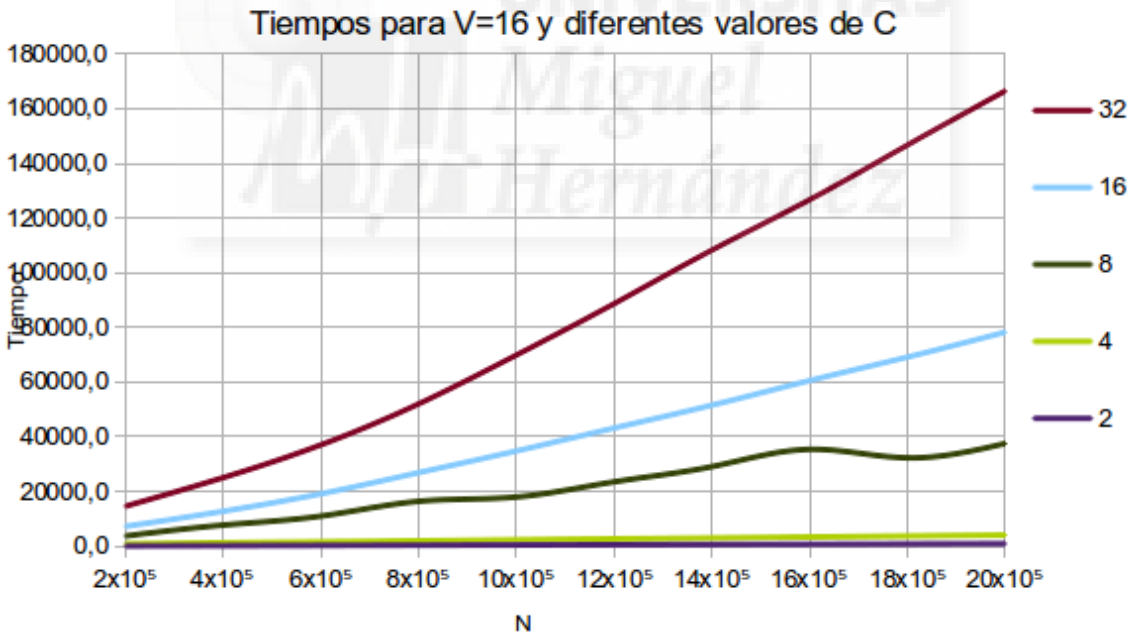


Figura 5.9: Tiempo frente a 'N' para 'V=16'

En la primera gráfica (figura 5.8) es muy notoria la diferencia entre la serie 'C=32' con respecto al resto, de lo que se puede deducir que para valores pequeños de 'V' (en ese caso 'V=2') un aumento en el número de columnas marca muy acusadamente un aumento de los tiempos de procesamiento. En cambio, la segunda gráfica (figura 5.9, 'V=16') los tiempos de procesamiento para el caso 'C=32' han aumentado relativamente poco mientras que las curvas 'V=16' y 'V=8' si presentan una diferencia algo más acusada. Es decir, aumentando 'V', pero no 'N' ni 'C', los tiempos de

ejecución parecen aproximarse a una cota máxima. Trivialmente, se puede comprobar como en la gráfica ‘V=2’ (figura 5.8), la curva ‘C=32’ pertenece al primer tramo de la función de la complejidad ($V^C = 2^{32} = 4.29x10^9$) mientras que el resto están en el tercer tramo ($V^{C+2} = 2^{18} = 262.144$). Análogamente, en la gráfica ‘V=16’ (figura 5.9) se observa que las curvas ‘C=16’ y ‘C=8’ han pasado al primer tramo de la función de complejidad ($V^C = 16^8 = 4.29x10^9$). La curva ‘C=4’ se ubica en el tramo 2 ($V^C = 16^4 = 65536$, $V^{C+2} = 16^8 = 4.29x10^9$); y, finalmente, la curva ‘C=2’ pertenece al tercer tramo de la función de complejidad ($V^{C+2} = 16^4 = 65536$).

5.2.3 Estudio 2: Análisis del tiempo frente a ‘C’ Parte 1: Comparativa por el número de registros ‘N’

Las series de datos para el análisis correspondiente a esta primera parte del segundo estudio se han extraído del cubo [NCV-T] como se indica en la figura 5.10. En esta ocasión, se han obtenido gráficas para comparar la evolución del tiempo para diferentes valores de ‘V’ fijado un valor concreto de ‘N’.

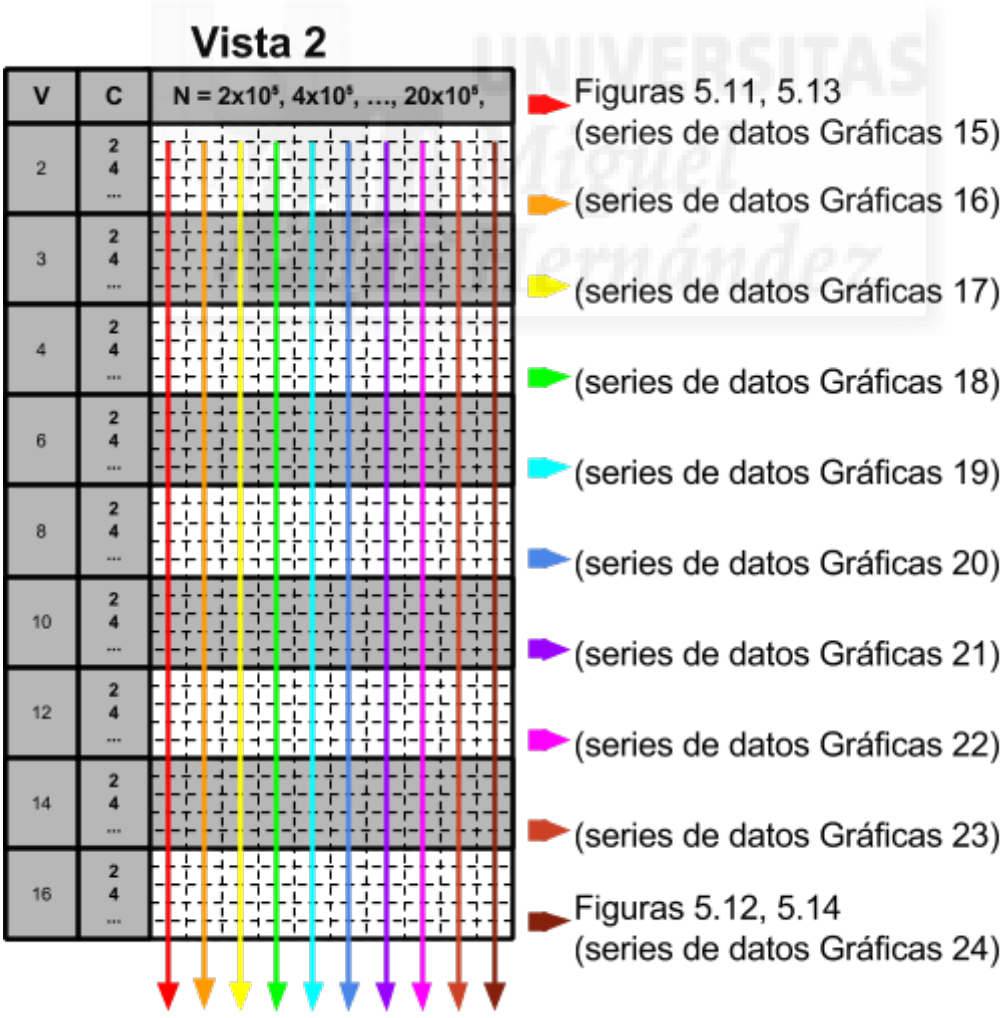


Figura 5.10: Origen de las series de datos para el “Estudio 2 - Parte 1”

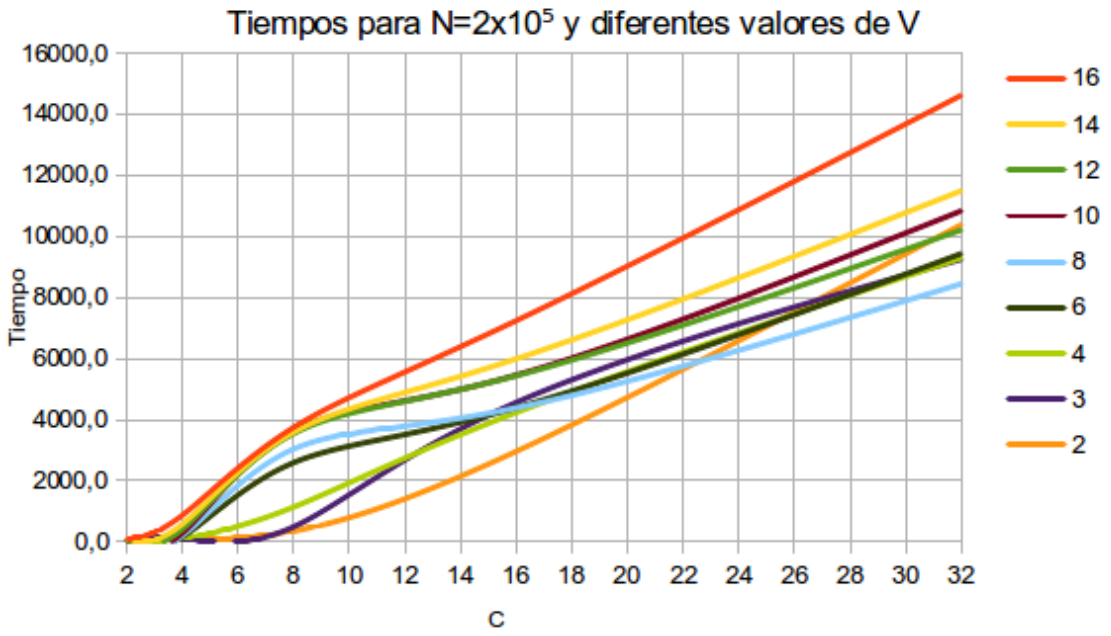


Figura 5.11: Tiempo frente a 'C' para 'N=2x10^{5}'

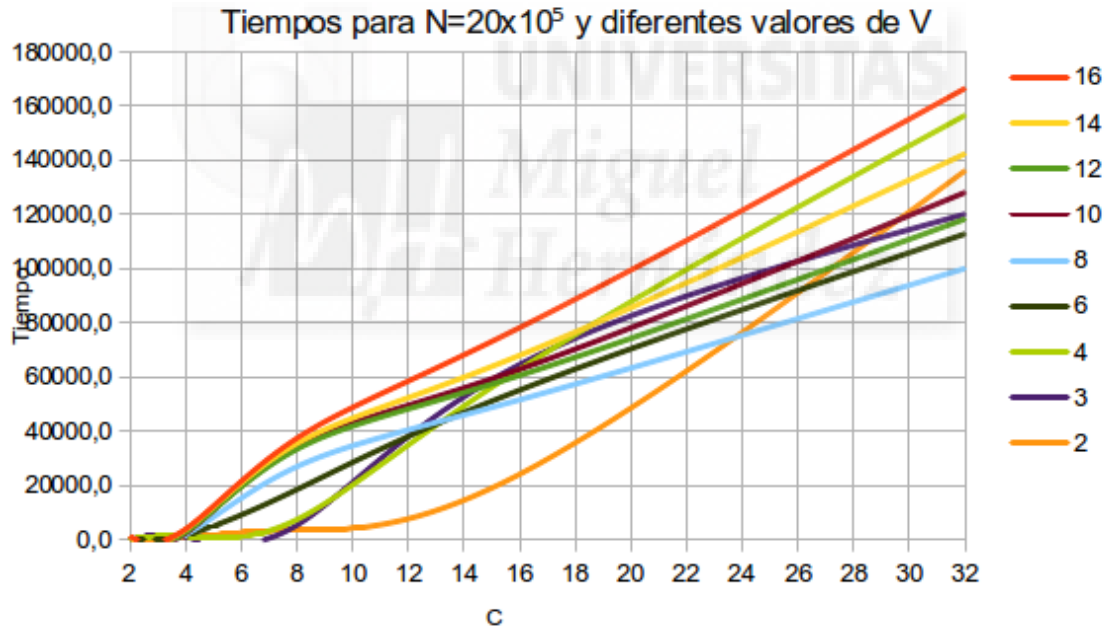


Figura 5.12: Tiempo frente a 'C' para 'N=20x10^{5}'

Estas gráficas, figuras 5.11 y 5.12, representan la evolución del tiempo en función de 'C' para los valores de 'N': '2x10^{5}' y '20x10^{5}' respectivamente. Se observa que el comportamiento es muy similar en ambos casos, con la salvedad de que hay una diferencia de orden '10' entre los tiempos de la primera gráfica y la segunda (misma diferencia que entre los valores de 'N' representados). Para los valores de 'C' considerados, de manera general, se observa un comportamiento no lineal al principio de la gráfica que tiende a estabilizarse linealmente a partir de un cierto valor de 'C', dicho valor de estabilización es mayor cuanto más pequeño es el 'V' representado. Las figuras 5.13 y 5.14 son, respectivamente, una sección de las anteriores 5.11 y 5.13.

Concretamente se quiere reproducir la parte inferior-izquierda de estas figuras, pero en esta ocasión, ambas gráficas presentan una misma escala para el eje de tiempos (eje Y), de esta forma se puede ver mejor, no solamente cómo evolucionan los tiempos de ejecución para cada valor de 'N', sino que también se aprecia mejor la diferencia entre las pendientes de las curvas para el caso de 'N=2x10^5' y el de 'N=20x10^5'.

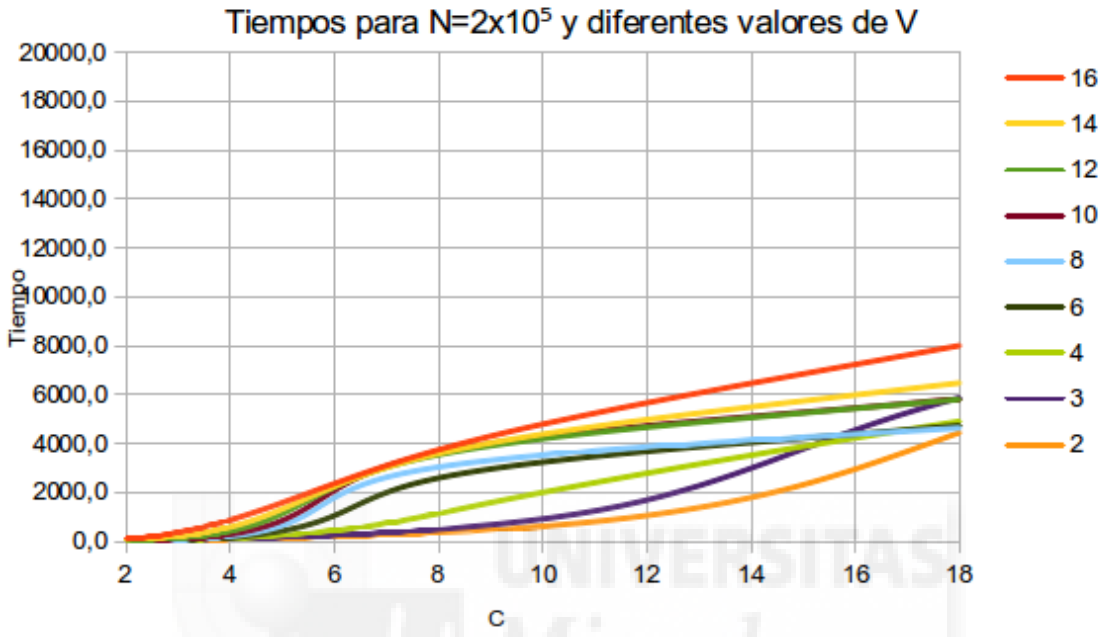


Figura 5.13: Tiempo frente a 'C' para 'N=2x10^5' (ampliación)

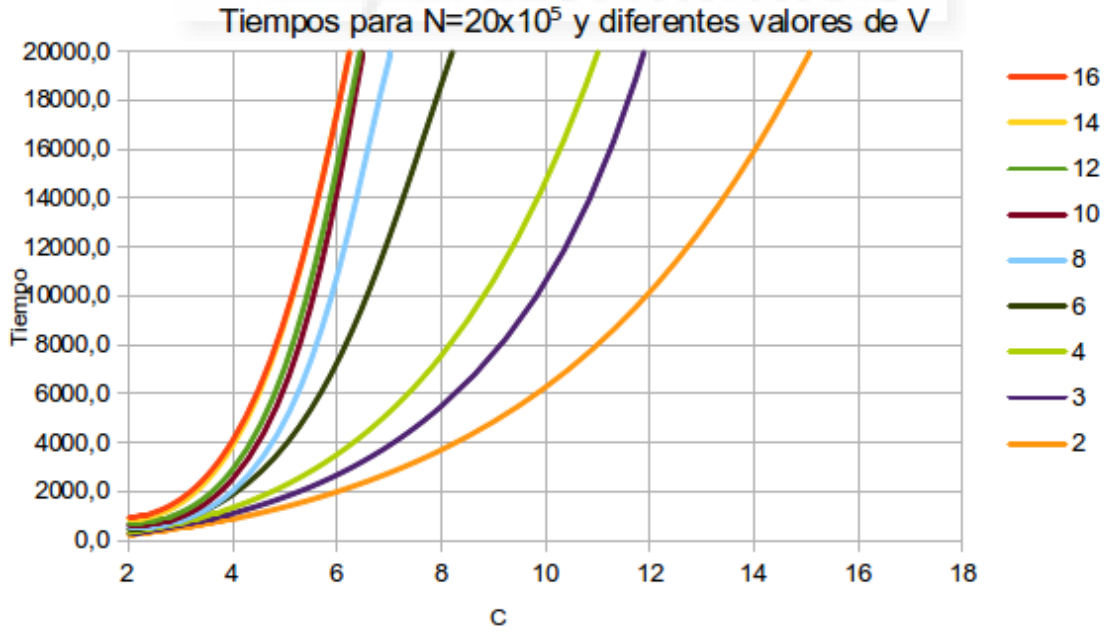


Figura 5.14: Tiempo frente a 'C' para 'N=20x10^5' (ampliación)

Como en casos anteriores, se comprueba a qué tramo de la función de complejidad teórica pertenece cada gráfica. Debido a que en esta ocasión son muchas las

combinaciones de ‘C’ y ‘V’ a considerar, se presenta la tabla 5.2 para indicar el posicionamiento por tramos de cada caso.

Tabla 5.2: Tramos función complejidad teórica

V	C	V^C	V^{C+2}	$N=2x10^5$	$N=20x10^5$
2	2	4	16	T-3	T-3
2	4	16	64	T-3	T-3
2	8	256	1024	T-3	T-3
2	16	65536	262144	T-2	T-3
2	32	4294967296	17179869184	T-1	T-1
3	2	9	81	T-3	T-3
3	4	81	729	T-3	T-3
3	8	6561	59049	T-3	T-3
3	16	43046721	387420489	T-1	T-1
3	32	1.85E+015	1.67E+016	T-1	T-1
4	2	16	256	T-3	T-3
4	4	256	4096	T-3	T-3
4	8	65536	1048576	T-2	T-3
4	16	4294967296	68719476736	T-1	T-1
4	32	1.84E+019	2.95E+020	T-1	T-1
6	2	36	1296	T-3	T-3
6	4	1296	46656	T-3	T-3
6	8	1679616	60466176	T-1	T-2
6	16	2821109907456	101559956668416	T-1	T-1
6	32	7.96E+024	2.87E+026	T-1	T-1
8	2	64	4096	T-3	T-3
8	4	4096	262144	T-2	T-3
8	8	16777216	1073741824	T-1	T-1
8	16	281474976710656	1.80E+016	T-1	T-1
8	32	7.92E+028	5.07E+030	T-1	T-1
10	2	100	10000	T-3	T-3
10	4	10000	1000000	T-2	T-3
10	8	100000000	10000000000	T-1	T-1
10	16	1.00E+016	1.00E+018	T-1	T-1
10	32	1.00E+032	1.00E+034	T-1	T-1
12	2	144	20736	T-3	T-3
12	4	20736	2985984	T-2	T-2
12	8	429981696	61917364224	T-1	T-1
12	16	1.85E+017	2.66E+019	T-1	T-1
12	32	3.42E+034	4.92E+036	T-1	T-1
14	2	196	38416	T-3	T-3
14	4	38416	7529536	T-2	T-2
14	8	1475789056	289254654976	T-1	T-1
14	16	2.18E+018	4.27E+020	T-1	T-1
14	32	4.74E+036	9.30E+038	T-1	T-1
16	2	256	65536	T-3	T-3
16	4	65536	16777216	T-2	T-2
16	8	4294967296	1099511627776	T-1	T-1
16	16	1,84E+019	4,72E+021	T-1	T-1
16	32	3.40E+038	8.71E+040	T-1	T-1

[T-1] - Tramo 1 [T-2] - Tramo 2 [T-3] - Tramo 3

En la tabla 5.2 se constata como para valores pequeños de ‘V’ y ‘C’ predomina el tramo 3 de la función de complejidad, mientras que, para valores cada vez mayores de esos parámetros, se tiende al tramo 1.

5.2.4 Estudio 2: Análisis del tiempo frente a ‘C’ Parte 2: Comparativa por el número de valores ‘V’

La figura 5.15 presenta cuál es el origen de datos para las series analizadas en este apartado del estudio. Se han generado gráficas que permiten comparar los tiempos de ejecución del algoritmo CREA para los diversos valores de ‘N’ fijando un determinado valor de ‘V’.

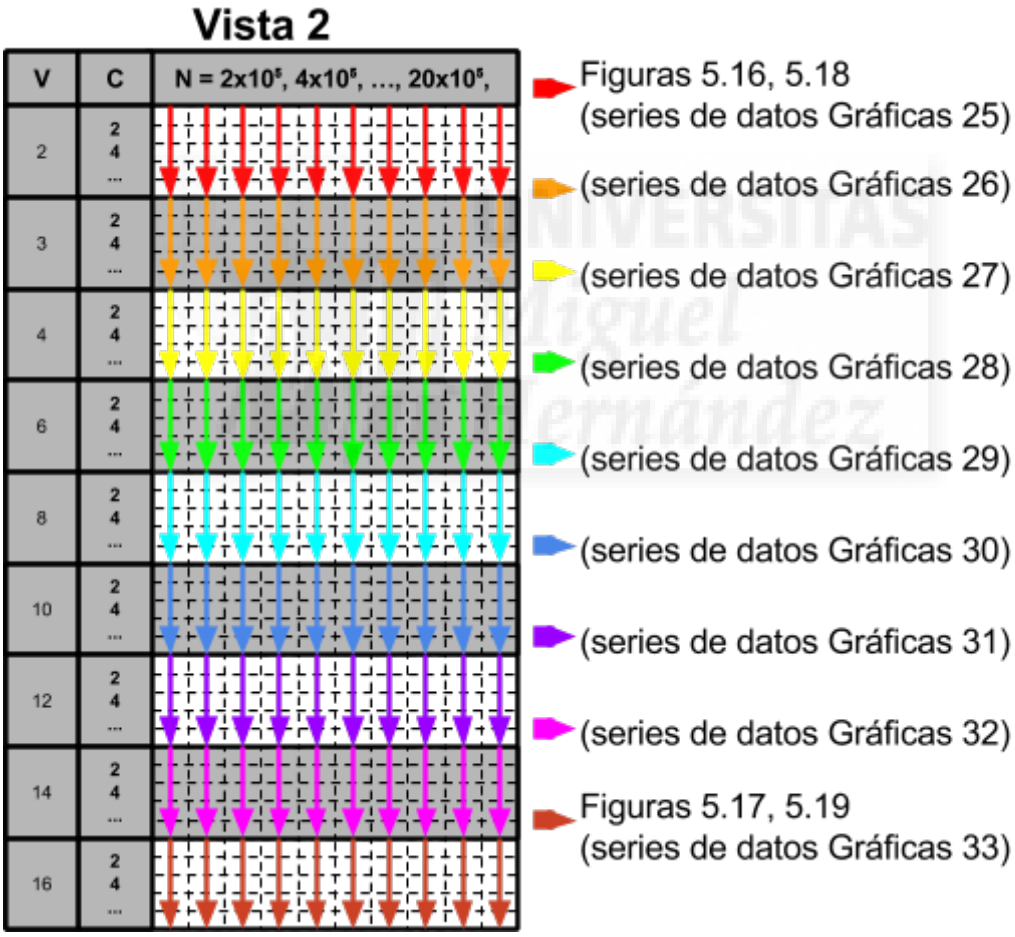


Figura 5.15: Origen de las series de datos para el “Estudio 2 - Parte 2”

En esta ocasión, las gráficas de las figuras 5.16 y 5.17 (‘V=2’ y ‘V=16’) muestran con bastante claridad el comportamiento de la evolución de los tiempos de ejecución de CREA en función de ‘C’. Dado que cada serie de datos representa un conjunto de datos con una misma cantidad de registros ‘N’, queda patente cómo a medida que se aumenta el número de columnas el tiempo crece, inicialmente con una pendiente pequeña (tramo

3). Luego se observa un comportamiento no lineal (tramo 2), para pasar a crecer nuevamente de forma lineal pero con una pendiente mayor (tramo 1).

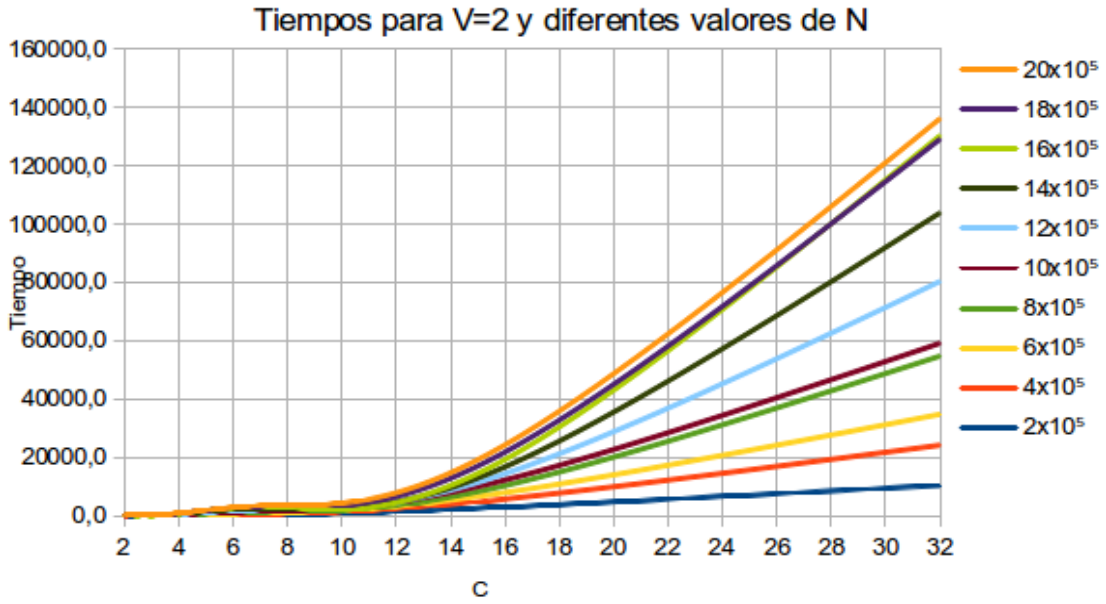


Figura 5.16: Tiempo frente a 'C' para 'V=2'

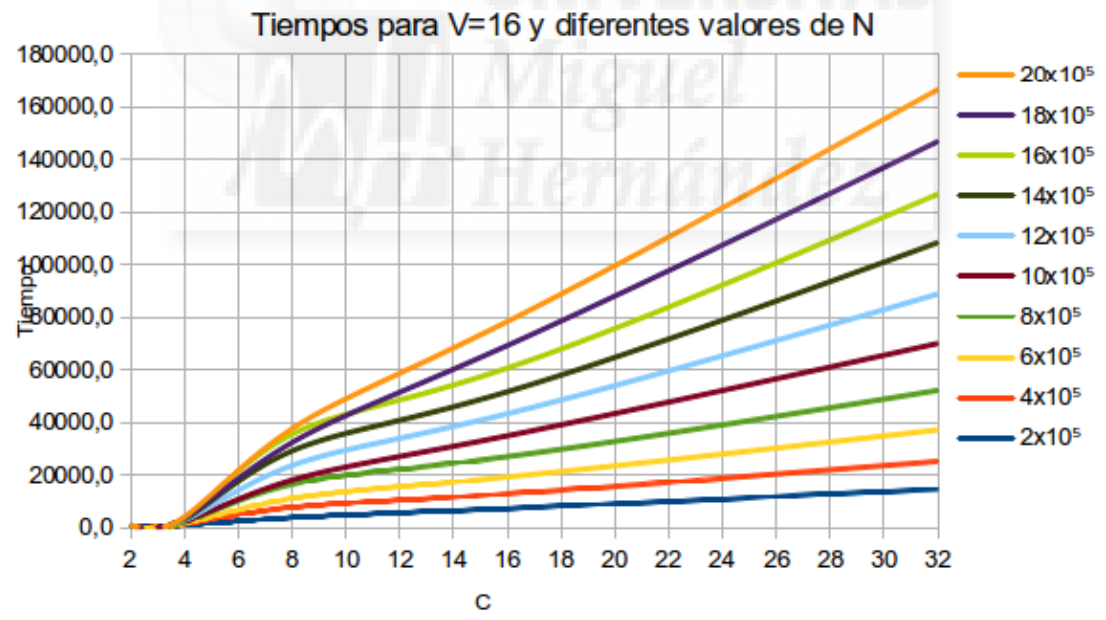


Figura 5.17: Tiempo frente a 'C' para 'V=16'

Como cabría esperar por el estudio teórico de la complejidad, el punto de cambio de tramo se adelanta en la segunda gráfica donde el valor de 'V' es mayor. Al igual que en el epígrafe anterior, se ha creído oportuno realizar una ampliación de la parte inferior de las figuras 5.16 y 5.17 (ver figuras 5.18 y 5.19 respectivamente) para apreciar con más detalle la evolución de las curvas de tiempos. La tabla 5.3 muestra los valores 'V^C' y 'V^{C+2}' para 'V=2' que se van a utilizar para señalar las fronteras de los tramos de la función de complejidad teórica que se pueden ver en la figura 5.18.

Tabla 5.3: Valores de V^C y V^{C+2} para V=2 y Tramos teóricos para cada valor de 'N' ($x10^5$)

C	2^C	2^{C+2}	N-2	N-4	N-6	N-8	N-10	N-12	N-14	N-16	N-18	N-20
2	**4**	**16**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
3	8	32	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
4	**16**	**64**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
5	32	128	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
6	64	256	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
7	128	512	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
8	**256**	**1024**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
9	512	2048	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
10	1024	4096	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
11	2048	8192	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
12	4096	16384	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
13	8192	32768	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
14	16384	65536	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
15	32768	131072	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
16	**65536**	**262144**	T2	T3	T3	T3	T3	T3	T3	T3	T3	T3
17	131072	524288	T2	T2	T3	T3	T3	T3	T3	T3	T3	T3
18	262144	1048576	T1	T2	T2	T2	T2	T3	T3	T3	T3	T3
19	524288	2097152	T1	T1	T2	T2	T2	T2	T2	T2	T2	T2
20	1048576	4194304	T1	T1	T1	T1	T1	T2	T2	T2	T2	T2
21	2097152	8388608	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1

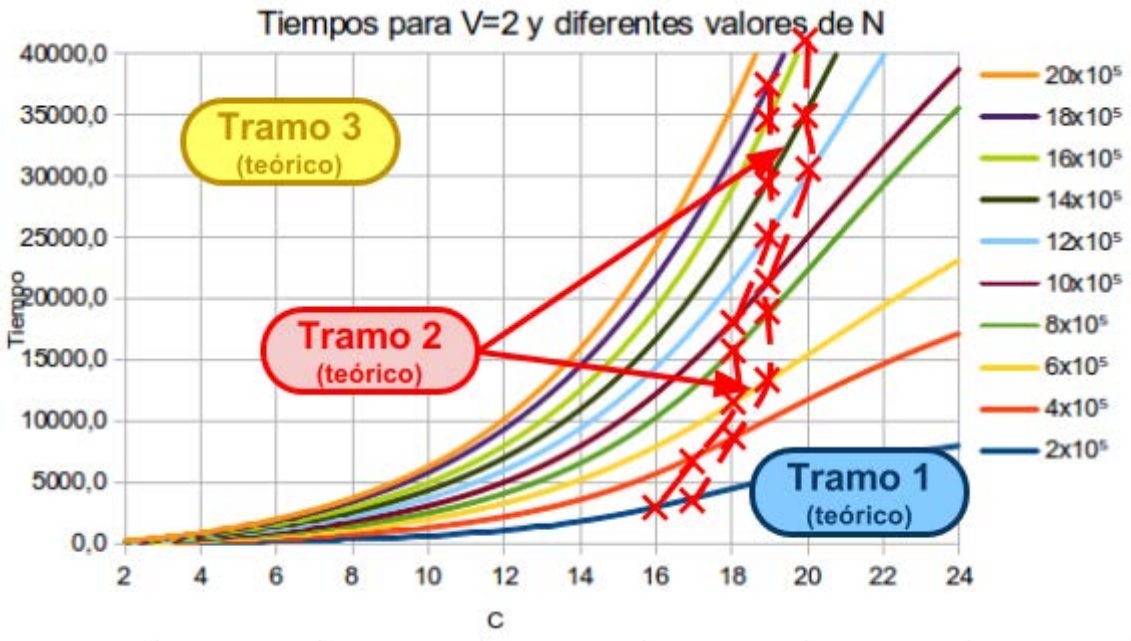


Figura 5.18: Tiempo frente a 'C' para 'V=2' (ampliado)

Del mismo modo que para el caso anterior, ahora para 'V=16', la tabla 5.4 muestra los valores 'V^{C}' y 'V^{C+2}' que identifican las fronteras de los tramos de la función de complejidad teórica que se pueden ver en la figura 5.19.

Tabla 5.4: Valores de V^{C} y V^{C+2} para V=16 y Tramos teóricos para cada valor de 'N' ($x10^5$)

C	16^{C}	16^{C+2}	N-2	N-4	N-6	N-8	N-10	N-12	N-14	N-16	N-18	N-20
2	**256**	**65536**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
3	4096	1048576	T2	T2	T2	T2	T2	T3	T3	T3	T3	T3
4	**65536**	**16777216**	T2	T2	T2	T2	T2	T2	T2	T2	T2	T2
5	1048576	268435456	T1	T1	T1	T1	T1	T2	T2	T2	T2	T2
6	16777216	4.29E+009	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
7	268435456	6.87E+010	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
8	**4.29E+009**	**1.01E+012**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
9	6.87E+010	1.75E+013	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
10	1.01E+012	2.81E+014	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
11	1.75E+013	4.50E+015	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
12	2.81E+014	7.21E+016	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
13	4.50E+015	1.15E+018	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
14	7.21E+016	1.84E+019	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
15	1.15E+018	2.95E+020	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
16	**1.84E+019**	**4.72E+021**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
17	2.95E+020	7.56E+022	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
18	4.72E+021	1.21E+024	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
19	7.56E+022	1.93E+025	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
20	1.21E+024	3.09E+026	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
21	1.93E+025	4.95E+027	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1

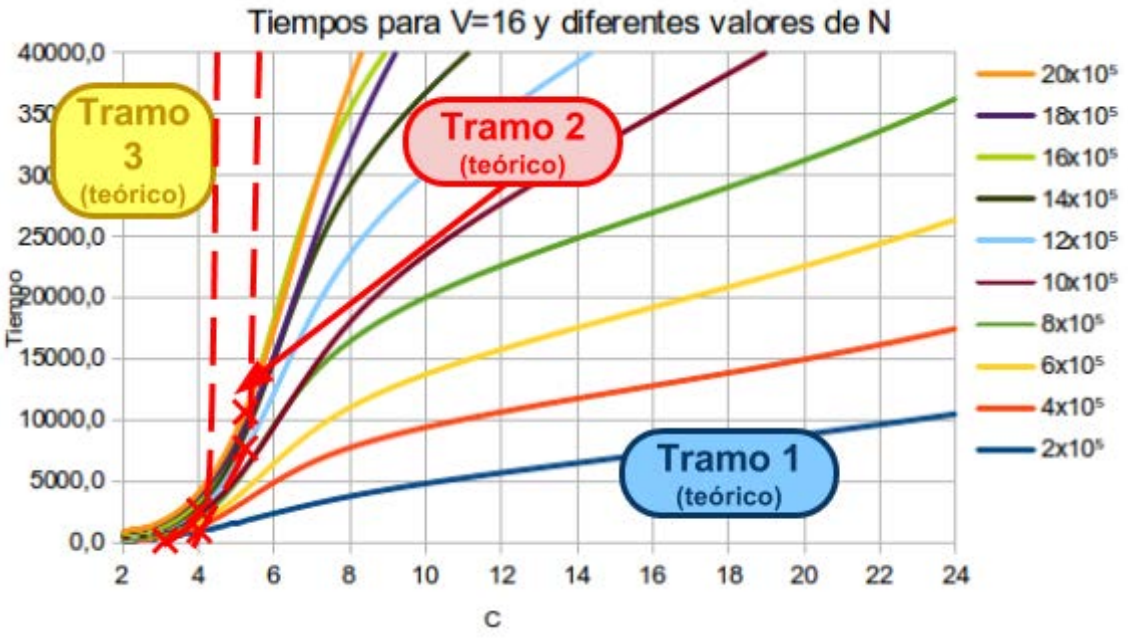


Figura 5.19: Tiempo frente a 'C' para 'V=16' (ampliado)

5.2.5 Estudio 3: Análisis del tiempo frente a ‘V’ Parte 1: Comparativa por el número de registros ‘N’

Las series de datos para el análisis correspondiente a esta primera parte del tercer estudio se han extraído del cubo [NCV-T] como se indica en la figura 5.20. Se han obtenido gráficas para comparar la evolución del tiempo para diferentes valores de ‘C’ fijado un valor concreto de ‘N’.

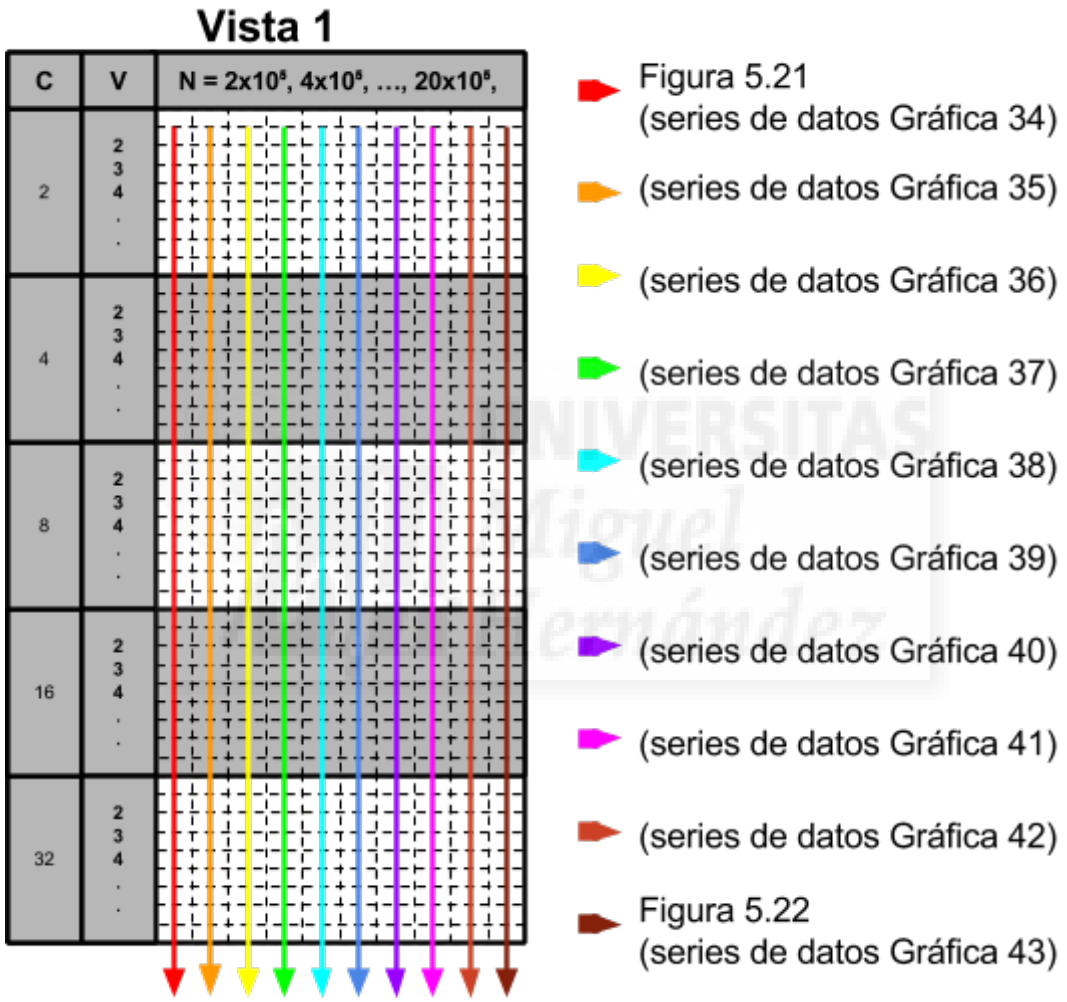


Figura 5.20: Origen de las series de datos para el “Estudio 3 - Parte 1”

Las gráficas obtenidas en este estudio reflejan que, para un valor de ‘N’ concreto, el parámetro ‘V’ incide en menor medida que el resto de variables en el aumento del tiempo, especialmente para valores grandes de ‘C’. En las siguientes gráficas (figuras 5.21 y 5.22) se observa como el tiempo de proceso para ‘C=32’ presenta una evolución errática, sin una tendencia clara, siendo más acusada para ‘N=20x10^5’, es decir, para el caso del conjunto de datos más grande.

Para los valores menores de ‘C’ el tiempo presenta una ligera tendencia creciente al aumentar ‘V’, dicha tendencia creciente es más clara cuanto más pequeño es el número

de columnas consideradas. En este caso puede concluirse que, para determinar la complejidad computacional del algoritmo CREA, para un mismo número de registros ‘N’, el parámetro ‘V’ va perdiendo importancia cuando aumenta el número de columnas del conjunto de datos considerado.

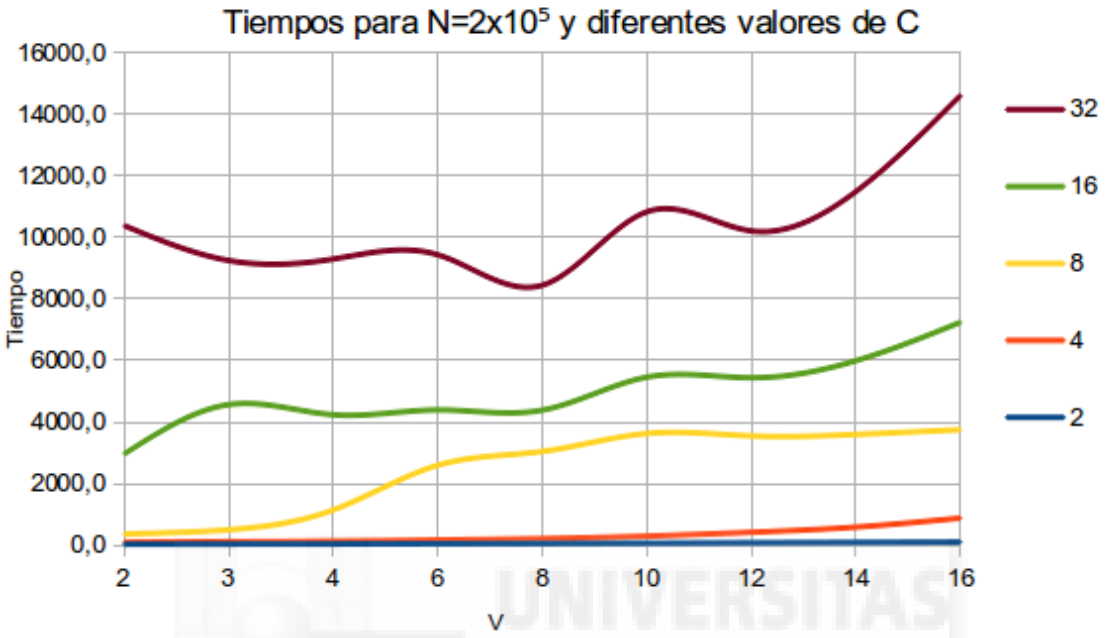


Figura 5.21: Tiempo frente a ‘V’ para ‘N=2x10^{5}’

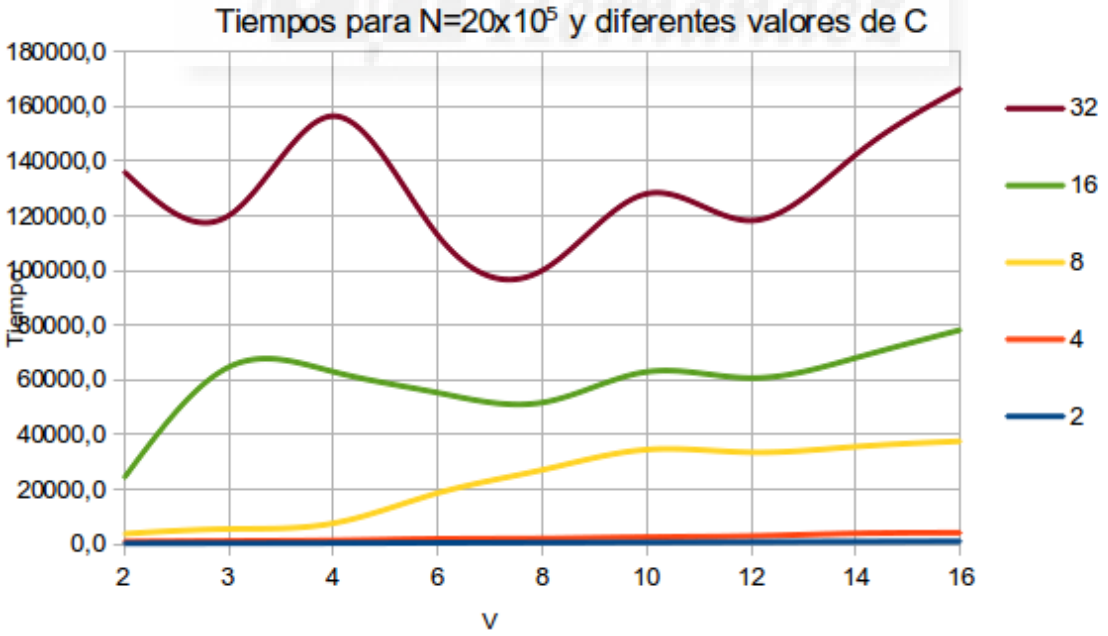


Figura 5.22: Tiempo frente a ‘V’ para ‘N=20x10^{5}’

En las gráficas anteriores (figuras 5.21 y 5.22), por motivos de escala, no se aprecia con claridad cómo crecen las curvas de tiempos para los valores más pequeños de ‘C’. La figura 5.23 es una ampliación de la parte inferior de la gráfica 5.21. En ella, ahora sí, se

observa como para los valores ‘C=2’, ‘C=4’ y ‘C=8’ las curvas de tiempos correspondientes sí presentan una tendencia creciente, claramente mayor para el valor mayor de ‘C’. De forma análoga, la figura 5.24 muestra una ampliación de la figura 5.22 que también confirma que para valores pequeños de ‘C’ los tiempos también son crecientes.

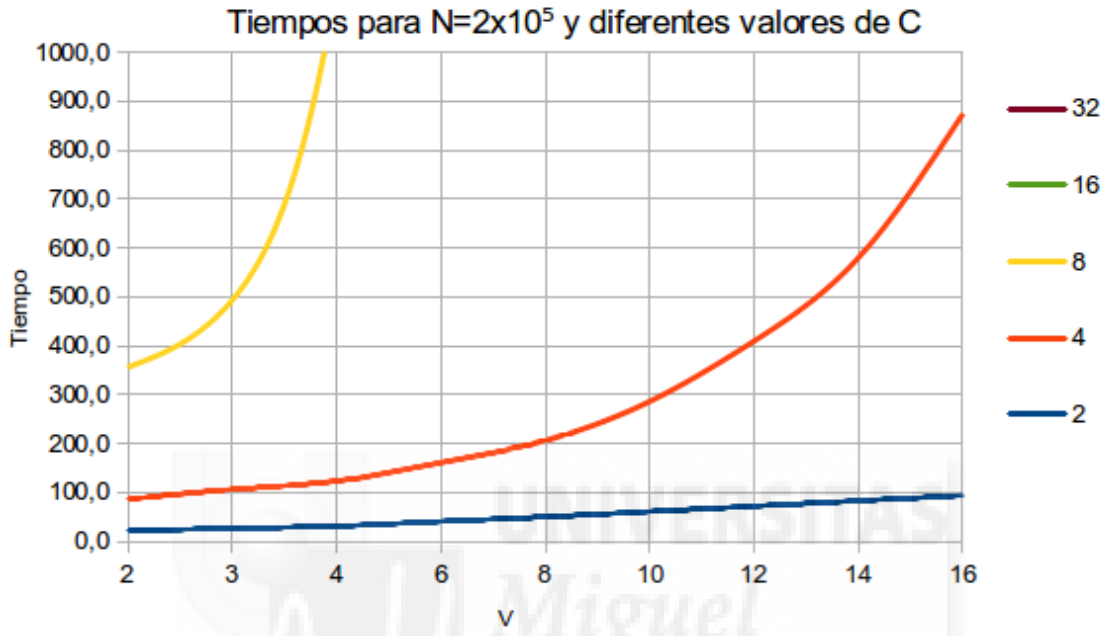


Figura 5.23: Tiempo frente a ‘V’ para ‘N=2x10^5’ (ampliación)

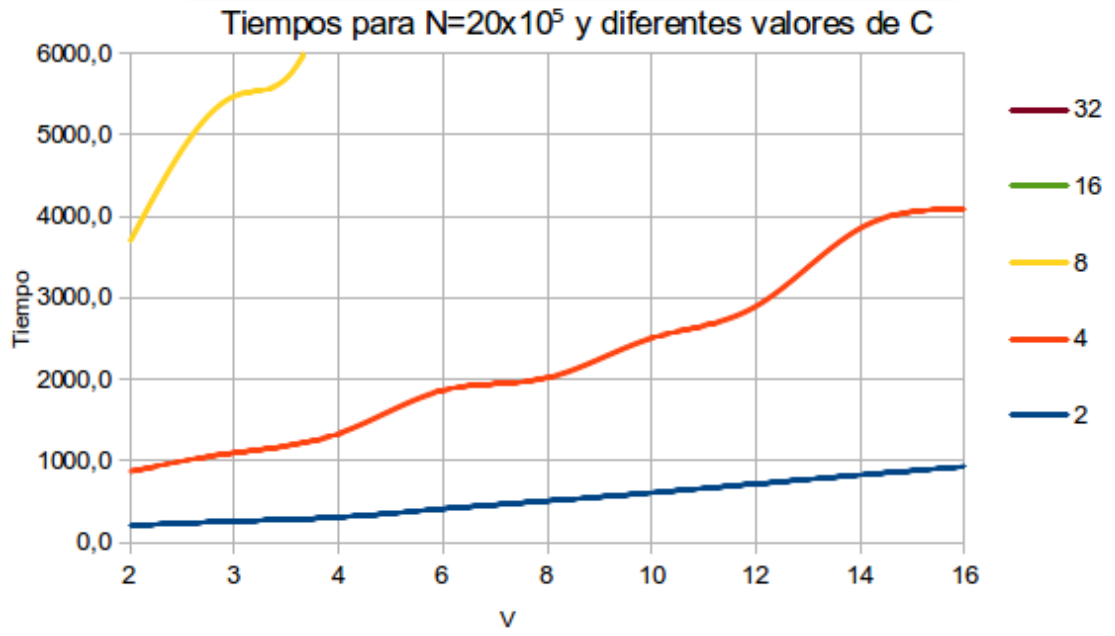


Figura 5.24: Tiempo frente a ‘V’ para ‘N=20x10^5’ (ampliación)

5.2.6 Estudio 3: Análisis del tiempo frente a ‘V’ Parte 2: Comparativa por el número de columnas ‘C’

En este último apartado del análisis multidimensional de los tiempos de ejecución del algoritmo CREA, se va a estudiar cómo evolucionan dichos tiempos en función de la variable ‘V’, fijando, en este caso, un valor de ‘C’ y obteniendo series de datos comparativas por el número de ejemplos del conjunto de datos. La figura 5.25 muestra esquemáticamente la forma en que se han tomado los datos del cubo [NCV-T] para representar las gráficas correspondientes a esta parte del análisis.

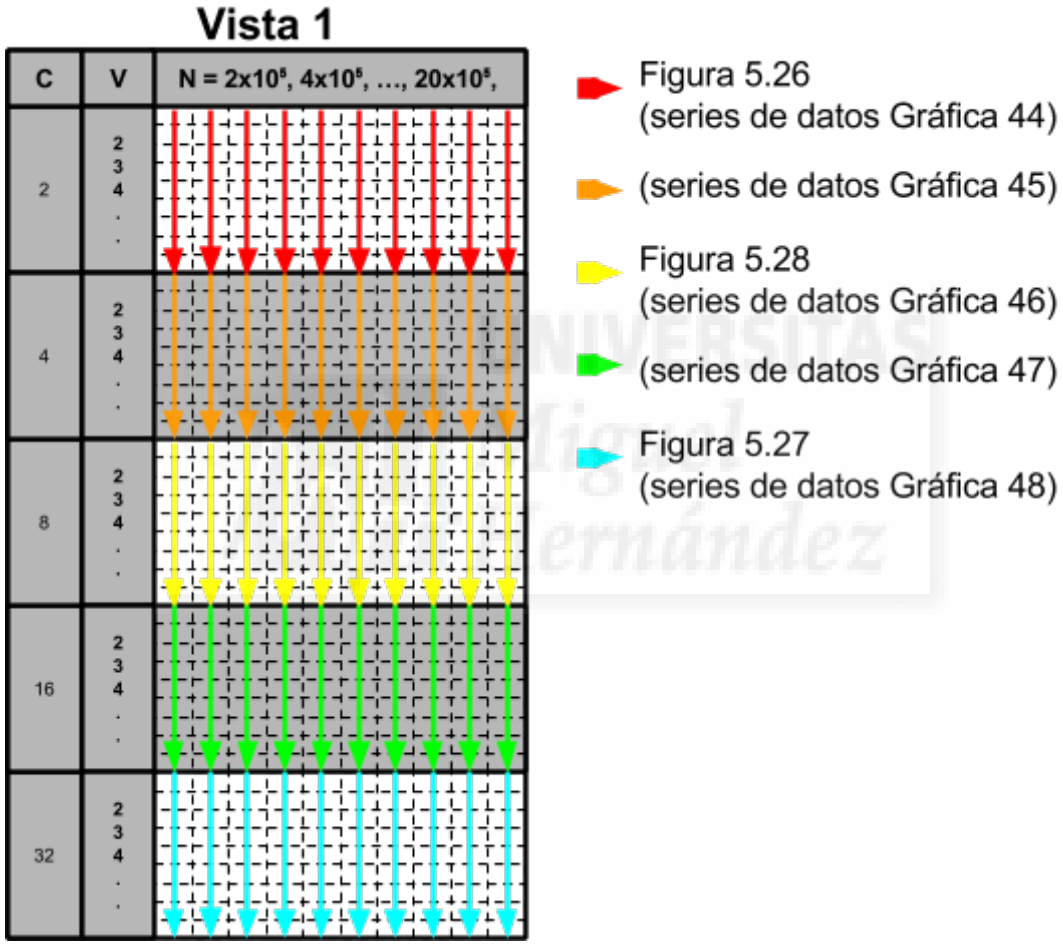


Figura 5.25: Origen de las series de datos para el “Estudio 3 - Parte 2”

En la figura 5.26, para ‘C=2’ (cantidad mínima de columnas) se observa que, para los diversos valores de ‘N’, la tendencia que presenta la evolución del tiempo es lineal en todos los casos. Se puede comprobar que todas las curvas de esta gráfica se caracterizan por hallarse en el tercer tramo de la función de complejidad ya que para el mayor valor de ‘V’ considerado (‘V=16’) y para el menor número de registros representado (‘N=2x10^5’) se cumple que ‘V^{C+2}< N’ (16^4 = 65536 < 200000).

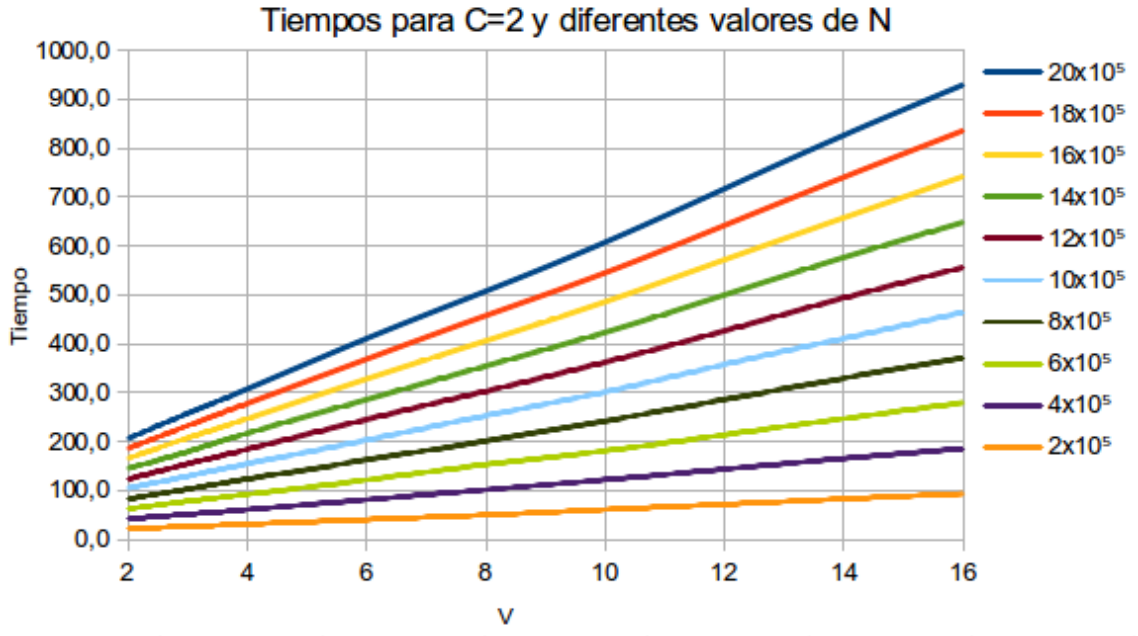


Figura 5.26: Tiempo frente a 'V' para 'C=2'

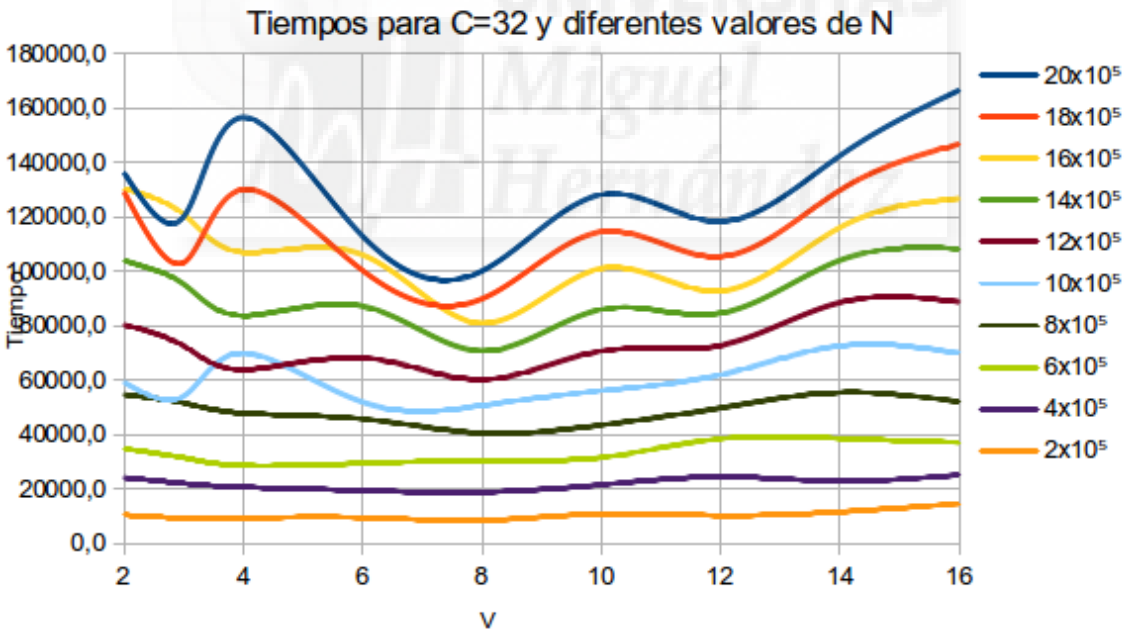


Figura 5.27: Tiempo frente a 'V' para 'C=32'

La figura 5.27, para 'C=32' (número máximo de columnas consideradas), ratifica la conclusión del epígrafe anterior, para valores grandes de 'C' la evolución del tiempo de ejecución es errática, sin una tendencia clara, presentando fluctuaciones más acusadas para los valores de 'N' mayores. Se confirma por tanto la deducción del apartado anterior: para valores pequeños de 'C', un aumento en el valor de 'V' produce aumentos en los tiempos de ejecución, en cambio, para valores grandes de 'C', el parámetro 'V' tiene menos influencia en el tiempo de ejecución.

Para ‘C=8’ se da un caso interesante en el que puede apreciarse cómo el tiempo de ejecución pasa por los tres tramos de la función de complejidad. La tabla 5.5 muestra la localización teórica de dichos tramos. La figura 5.28 muestra la evolución del tiempo de ejecución en función de ‘V’ para conjuntos de datos de ocho columnas. En este caso se puede apreciar como al principio de la gráfica se produce un cambio de tendencia acorde con la definición teórica por tramos de la función de complejidad.

Tabla 5.5: Valores de V^C y V^{C+2} para C=8 y Tramos para cada valor de ‘N’ ($x10^5$)

V	$V^C = V^8$	$V^{C+2} = V^{10}$	N-2	N-4	N-6	N-8	N-10	N-12	N-14	N-16	N-18	N-20
2	**256**	**1024**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
3	**6561**	**59049**	T3	T3	T3	T3	T3	T3	T3	T3	T3	T3
4	**65536**	**1048576**	T2	T2	T2	T2	T2	T3	T3	T3	T3	T3
5	390625	9765625	T1	T2	T2	T2	T2	T2	T2	T2	T2	T2
6	**1679616**	**60466176**	T1	T1	T1	T1	T1	T1	T1	T1	T2	T2
7	5764801	282475249	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
8	**16777216**	**1.07×10^9**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
9	43046721	3.47×10^9	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
10	**100000000**	**1.00×10^{10}**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
11	214358881	2.59×10^{10}	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
12	**429981696**	**6.19×10^{10}**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
13	815730721	1.38×10^{11}	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
14	**1.48×10^9**	**2.89×10^{11}**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
15	2.56×10^9	5.77×10^{11}	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1
16	**4.29×10^9**	**1.10×10^{12}**	T1	T1	T1	T1	T1	T1	T1	T1	T1	T1

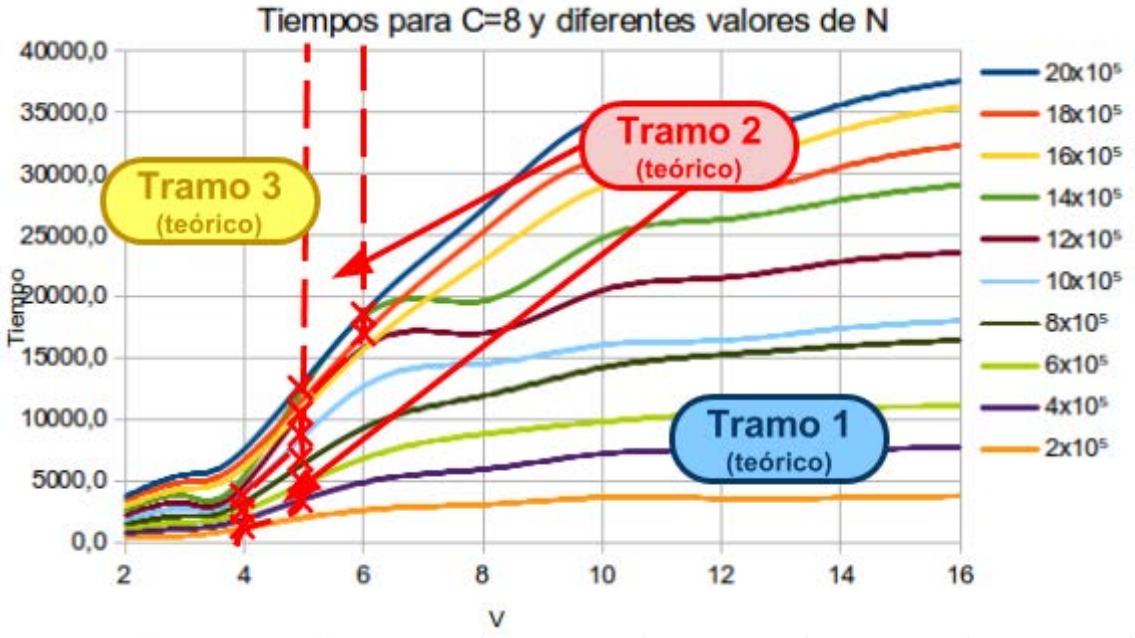


Figura 5.28: Tiempo frente a ‘V’ para ‘C=8’

5.2.7 Discusión sobre el análisis

A lo largo de todo el epígrafe 5.2 se ha realizado un análisis pormenorizado de la evolución de los tiempos de ejecución del algoritmo CREA en función de los parámetros 'N', 'C' y 'V'. A continuación se pasa discutir de forma conjunta y sintetizada los resultados obtenidos en cada una de las partes del citado análisis.

Con respecto a 'N', queda patente que este parámetro influye linealmente en los tiempos de ejecución. En todos los casos un aumento en el tamaño de la muestra produce aumentos proporcionales en los tiempos de ejecución, si bien para un número elevado de columnas se producen algunas fluctuaciones en las gráficas, la tendencia observada sigue siendo linealmente creciente. Igualmente se observa como la pendiente de las líneas de tiempo cambia según los valores de 'C' y 'V' considerados. Este cambio se hace especialmente evidente en la figura 5.6 en la que se percibe una diferencia notable en las pendientes de las curvas que pertenecen al primer tramo de la función de complejidad, respecto a la que se corresponde con el tercer tramo. Por otra parte, las pruebas empíricas indican que a medida que aumenta el valor de 'C', el parámetro 'V' influye menos en el incremento de tiempo de proceso.

Considerando el parámetro 'C', se ve como al principio de las curvas (valores pequeños de 'C') el tiempo de ejecución presenta comportamientos lineales con baja pendiente (tercer tramo), después se aprecia un comportamiento no lineal (segundo tramo), para finalmente volver a presentar un comportamiento lineal con pendiente mayor que al principio (tercer tramo). Para valores pequeños de 'V', las partes de las curvas pertenecientes a los tramos tercero y segundo ocupan un espacio mayor en las gráficas. En cambio, para valores grandes de 'V' esos mismos tramos se encuentran más próximos al principio de las gráficas, siendo el tercer tramo el que ocupa casi la totalidad de cada una de ellas. Si bien se ve cómo al aumentar 'V' los tiempos de ejecución aumentan más deprisa al principio, a medida que aumenta 'C' los incrementos de 'V' son cada vez menos influyentes en el tiempo.

En el estudio realizado sobre el parámetro 'V' se confirma el hecho de que los incrementos en este parámetro producen un incremento en el tiempo de ejecución pero solamente cuando los valores de 'C' son bajos o intermedios; en cambio, para valores de 'C' grandes el parámetro 'V' parece no influir en absoluto en los tiempos de ejecución, confirmándose la anterior observación acerca de la influencia de 'V' para los valores superiores de 'C'. Es especialmente interesante el caso de conjuntos de datos de 8 columnas ('C=8', figura 5.28) en el que se puede ver como las curvas de tiempos evolucionan pasando por los tres tramos de la función de complejidad, confirmándose una vez más la coherencia entre el estudio teórico del coste computacional del algoritmo CREA y el presente análisis empírico.

Queda añadir que en algunas de las gráficas donde se han superpuesto los tramos teóricos de la función de complejidad, no siempre coinciden exactamente con los tramos que en dichas gráficas presentan la evolución temporal empíricamente observada. Debe tenerse en cuenta que la función de complejidad calculada en el capítulo 3 (expresión 3.12), es una aproximación asintótica, no un valor exacto del tiempo que tarda el algoritmo CREA en ejecutarse. Además, esta función se ha calculado considerando el peor escenario posible en el que el conjunto de datos de entrada para CREA genera un número máximo de reglas y todas ellas están totalmente expandidas, cosa que no es normal que pase en un caso real. Por estas razones es de esperar que las cotas o márgenes que delimitan los tramos de la función de complejidad teórica no siempre coincidan totalmente con los tramos observados a partir de las mediciones sobre los conjuntos de datos simulados. La naturaleza concreta del conjunto de datos que se esté analizando en cada caso (independientemente de los valores 'N', 'C' y 'V') siempre influirá en el tiempo de ejecución del algoritmo. No obstante, sí se observa en la experiencia computacional la existencia de estos tres tramos que, aunque no coincidan exactamente con los obtenidos teóricamente, se aproximan suficientemente a estos. Nótese también que las magnitudes 'N', 'C' y 'V' son discretas y las gráficas presentadas en este apartado se han dibujado de forma continua, mediante interpolación, con el objetivo de que fuese más fácilmente apreciable la tendencia que en cada caso presenta cada curva, aunque dicha interpolación es siempre una aproximación.

En el capítulo 2 de esta tesis (epígrafe 2.2, objetivo 5) se planteó verificar si se cumplía la siguiente hipótesis: Dados dos conjuntos de datos con igual número de ejemplos, si se cumple que la potencia 'V^C' coincide en ambos casos, los tiempos de ejecución del algoritmo CREA para dichas muestras coinciden o son muy parecidos. La tabla 5.1 muestra el valor de 'V^C' para los conjuntos de datos considerados en el análisis empírico, se encuentran coloreados del mismo color aquellos casos en los que el valor 'V^C' coincide. Para verificar o rechazar empíricamente la hipótesis formulada habrá que prestar especial atención a los tiempos de ejecución obtenidos en los conjuntos de datos indicados.

Tabla 5.6: Valores de V^C para los conjuntos de datos simulados

V^C	C = 2	C = 4	C = 8	C = 16	C = 32
V = 2	4	16	256	65536	4294967296
V = 3	9	81	6561	43046721	1.85×10^{15}
V = 4	16	256	65536	4294967296	1.84×10^{19}
V = 6	36	1296	1679616	2.82×10^{12}	7.96×10^{24}
V = 8	64	4096	16777216	2.81×10^{14}	7.92×10^{28}
V = 10	100	10000	100000000	1.00×10^{16}	1.00×10^{32}
V = 12	144	20736	429981696	1.85×10^{17}	3.42×10^{34}
V = 14	196	38416	1475789056	2.18×10^{18}	4.74×10^{36}
V = 16	256	65536	4294967296	1.84×10^{19}	3.40×10^{38}

La tabla 5.7 muestra una parte de los tiempos de ejecución del algoritmo CREA, resaltando con igual color los casos en los que el valor 'V^C' coincide.

Tabla 5.7: Tiempos de ejecución según valor V^C

V	C	N= $2x10^5$	N= $4x10^5$	N= $6x10^5$	N= $8x10^5$	N= $10x10^5$	N= $12x10^5$	N= $14x10^5$	N= $16x10^5$	N= $18x10^5$	N= $20x10^5$
2	2	21.9	42.2	62.5	82.8	104.7	123.4	145.3	165.7	185.9	206.2
2	4	85.9	173.4	259.4	348.4	435.9	521.9	609.4	693.7	782,8	870.3
2	8	356.2	742.2	1143.8	1454.7	1803.1	2175.0	2559.4	2921.8	3400,0	3693.7
2	16	2961.0	5690.6	7904.7	10278.1	12220.3	14364.1	16687.5	19207.8	21746.8	24231.3
2	32	10381.2	24142.2	34725.0	54681.2	59109.4	80293.8	103779.7	130282.8	129020.3	136096.9
4	2	31.2	60.9	92.2	123.5	154.7	184.4	217.2	246.9	278.1	307.8
4	4	123.4	251.5	384.3	510.9	671.9	778.1	940.7	1082.8	1203.1	1334.3
4	8	1139.1	1871.9	2548.4	3243.8	3995.3	4635.9	5386.0	6084.4	6714.0	7543.8
4	16	4231.3	9300.0	17284.4	20937.5	29150.0	39946.9	52575.0	46112.5	51581.3	62971.9
4	32	9296.9	20646.9	28576.5	47801.6	69915.7	63828.1	83643.7	106926.6	130068.7	156540.6
16	2	93.7	185.9	279.7	371.9	464.1	556.3	648.4	742.2	835.9	929.6
16	4	873.5	1292.2	1623.4	1971.9	2348.4	2704.7	3029.7	3385.9	3740.6	4081.3
16	8	3746.9	7729.7	11017.2	16425.0	18003.1	23561.0	29109.4	35428.2	32306.3	37560.9
16	16	7239.1	12782.8	19184.3	26917.2	34836.0	43226.5	51579.7	60587.5	69168.8	78279.7
16	32	14615.6	25098.4	37075.0	52078.2	69934.3	88725.0	108309.4	126728.1	146712.5	166485.9

V^C=16 V^C=256 V^C=65536 V^C=$4.29x10^9$ V^C=$1.84x10^{19}$

A la vista de los datos puede concluirse que la hipótesis planteada debe ser rechazada, ya que en ningún caso se aprecia que los tiempos de ejecución coincidan o se aproximen para valores iguales de 'V^C'. El comportamiento observado resulta ser que, para dos conjuntos de datos 'E_1' y 'E_2' caracterizados con los atributos 'N_1', 'C_1' y 'V_1' el primero, y 'N_2', 'C_2' y 'V_2' el segundo, cumpliéndose que '$N_1=N_2$' y '$V_1^{C1}=V_2^{C2}$', se tiene que los tiempos de ejecución son mayores cuando 'C' es mayor. Gráficamente, las figuras 5.29 y 5.30 (para '$V^C=256$' y '$V^C=4.29x10^9$' respectivamente) corroboran este hecho (en el anexo II se pueden encontrar otras gráficas que también lo confirman).

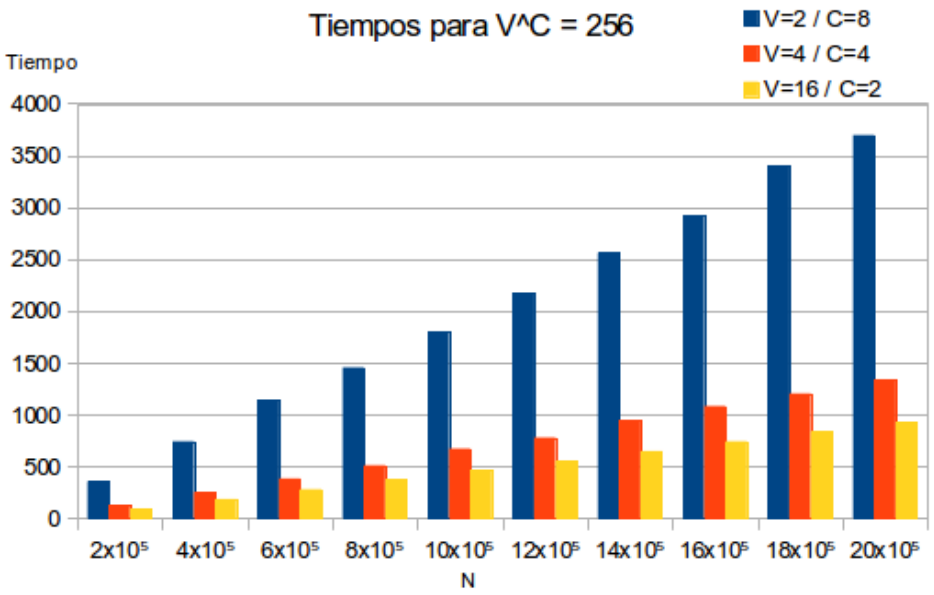


Figura 5.29: Comparativa de tiempos para conjuntos con V^C=256

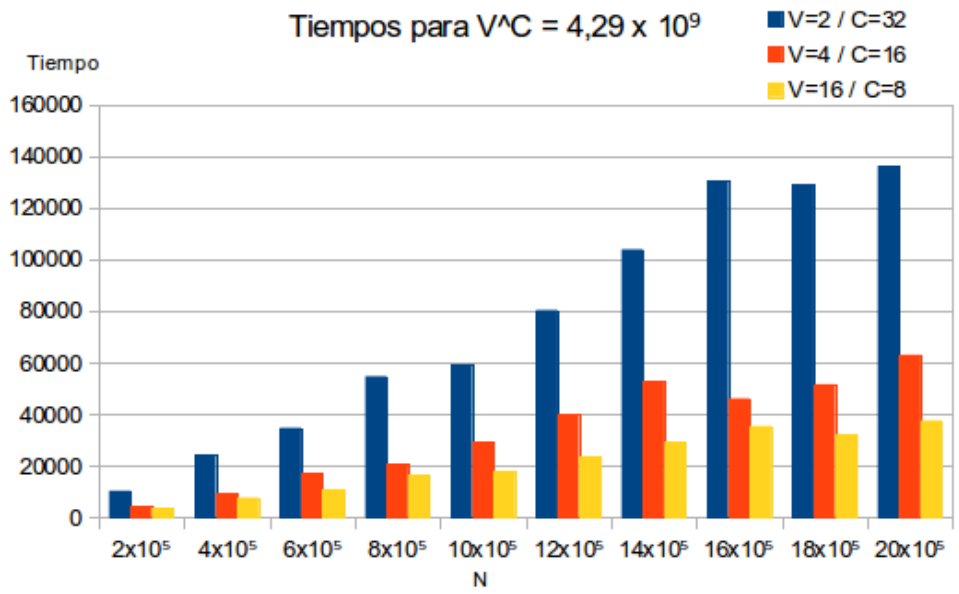


Figura 5.30: Comparativa de tiempos para conjuntos con V^C=4.29 x 10^9

5.3. Resultados de CREA sobre datos reales venta B2B

Para verificar la eficacia del algoritmo CREA y comprobar que, efectivamente, permite generar modelos válidos en el ámbito estratégico-empresarial, se ha realizado un segundo trabajo basado en la utilización de datos reales provenientes de las ventas realizadas por la empresa MUSTANG. Este conjunto de datos ya fue descrito en el capítulo 4 (epígrafe 4.3), donde también se comentó el preprocesamiento realizado con el mismo. A modo de resumen se vuelve a transcribir la tabla 4.13 que sintetiza las características del conjunto final de datos procesados por el algoritmo CREA tras el preprocesamiento.

Repetición de la tabla 4.13: Descripción del conjunto de datos empresariales

Número de registros: N	N = 423085
Número de columnas: C	C = 10
Número de valores por columna: V	V1 = |Sem-Ped| = 52 V2 = |Marca| = 7 V3 = |PrecioParD| = 5 V4 = |Temporada| = 14 V5 = |Categoría| = 3 V6 = |Estilo| = 4 V7 = |Tipo| = 14 V8 = |Corte| = 25 V9 = |Rasgo| = 29 V10 = |UnidadesD| = 6

Nótese que en este caso, por tratarse de datos reales (no creados de forma sintética), el parámetro 'V' no tiene el mismo valor para todos los atributos del conjunto de datos, sino que más bien es diferente para (casi) todos ellos, por lo que en este caso, el valor 'V^C' se asimilará al producto de todos los valores 'V_i' de todos los atributos considerados.

$$V^C \approx \prod_{i=1}^{10} Vi \qquad \text{(Expresión 5.3)}$$

Como también se determinó en el epígrafe 4.3, el objetivo empresarial es caracterizar la cantidad de productos vendidos en función del resto de atributos seleccionados, por tanto, la variable objetivo en este estudio es '*UnidadesD*', es decir, este atributo va a ser el consecuente de las reglas que se van a generar y el resto formarán parte del antecedente.

5.3.1. Extracción de características más relevantes

Antes de generar ningún conjunto de reglas se ha realizado un estudio previo para determinar cuáles de los atributos del antecedente o qué subconjunto de ellos está mejor correlacionado con la variable consecuente. Para ello se ha utilizado una variante de la métrica 'ACI' ('*Attribute Correlation Index*' o '*Índice de Correlación de Atributos*') que se calcula en el método RBS (ver epígrafe 2.2.1), y que ya fue utilizada con resultados satisfactorios en el trabajo de Zaragozí (2011). La variante de esta métrica, a la que se ha denominado 'WACI' ('*Weighted Attribute Correlation Index*' o '*Índice de Correlación de Atributos Ponderado*') consiste en aplicar un peso 'w_i' a cada una de las regiones con reglas significativas identificadas en el método RBS. La fórmula para el cálculo de esta métrica es la indicada en la expresión 5.4.

$$WACI(Rr,w_1,w_2,w_3) = \frac{(w_1 \cdot |REG\text{-}1| + w_2 \cdot |REG\text{-}2| + w_3 \cdot |REG\text{-}3|) / | R |}{w_1 \cdot area(REG\text{-}1) + w_2 \cdot area(REG\text{-}2) + w_3 \cdot area(REG\text{-}3)} \qquad \text{(Expresión 5.4)}$$

Esta fórmula para calcular la métrica 'WACI' deriva de la expresión 2.10 y como puede observarse aplica unos pesos 'w_1', 'w_2' y 'w_3' al número de reglas y al área de cada una de las regiones 'REG-1', 'REG-2' y 'REG-3'. El objetivo de esta nueva métrica es permitir al analista de los datos dar mayor o menor importancia a unas regiones frente a otras. Nótese que si se da el valor '1' a los tres pesos, la nueva métrica 'WACI' coincide con el valor 'ACI', es decir, se cumple la igualdad de la expresión 5.5.

$$WACI(Rr, 1, 1, 1) = ACI(Rr) \qquad \text{(Expresión 5.5)}$$

En este estudio se consideró que las reglas de la región 'REG-3' no eran importantes para el análisis por lo que se optó por usar los siguientes pesos para todo el análisis: 'w_1=1', 'w_2=1' y 'w_3=0'.

Dado que el número de atributos a considerar en el antecedente no es muy grande, solamente 9, se decidió probar un método de fuerza bruta que ejecuta todas las posibles combinaciones de antecedentes. Las reglas con un antecedente mínimo son de la forma:

$$\{\text{Atr-i}\} \rightarrow \{\text{UnidadesD}\} \qquad \text{(Expresión 5.6)}$$

Donde 'Atr-i' puede ser cualquiera de las variables que pueden formar parte del antecedente. Por tanto se generarán 9 modelos diferentes con esta forma. Por otra parte, las reglas con el mayor antecedente posible tiene la forma:

$$\{\text{Atr-1, Atr-2, …, Atr-9}\} \rightarrow \{\text{UnidadesD}\} \qquad \text{(Expresión 5.7)}$$

Donde la tupla '{Atr-1, Atr-2, …, Atr-9}' estará formada por las 9 variables antecedentes. En este caso se generará un único modelo de reglas de clasificación. Considerando estos casos y todos los posibles casos intermedios, reglas que tengan en el antecedente 2, 3, 4, 5, 6, 7 y 8 variables, en total se han considerado 2^9-1=511 modelos. En la tabla 5.8 se muestran el ranking de las 20 combinaciones de antecedentes que mejor valor de 'WACI' obtuvieron. A raíz del estudio realizado se pueden extraer las siguientes conclusiones:

1. Individualmente, las variables '*Estilo*' y '*Marca*' presentan la mejor correlación con la cantidad de unidades pedidas. Ambas tienen una incidencia casi idéntica, siendo el '*Estilo*' ligeramente superior (por unas pocas milésimas).

2. Consideradas globalmente, es decir, en los 20 resultados obtenidos, las dos variables siguen teniendo una importancia similar, ya que ambas aparecen en 7 de las 20 mejores combinaciones.

3. Precisamente, conjuntamente '*Marca*' y '*Estilo*' son el tercer mejor resultado, que apenas difiere en 6,5 y 7 centésimas de los dos mejores resultados.

4. Las variables '*Categoría*' y '*PrecioParD*' (precio par discretizado) son, respectivamente, la tercera y cuarta en importancia ya que aparecen en las combinaciones cuarta y quinta, y conjuntamente en la sexta. Además, globalmente se repiten 7 y 5 veces respectivamente.

5. Las variables '*Rasgo*' y '*Temporada*' parecen tener una importancia media/baja ya que no aparecen hasta la octava y novena combinación.

6. Los atributos '*Tipo*' y '*Corte*' presentan una correlación menor que los anteriores, ya que no aparecen hasta las combinaciones decimocuarta y vigésima.

7. El campo '*SemPed*' (semana del pedido) no aparece en ninguna de las 20 combinaciones consideradas.

Tabla 5.8: Mejores combinaciones de variables según 'WACI' para w_1=1, w_2=1 y w_3=0

	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	Tipo	Corte	Rasgo	WACI	∇ WACI
#1						**Estilo**				0.4039	---
#2		**Marca**								0.3983	0.0056
#3		**Marca**				**Estilo**				0.3326	0.0657
#4		**Marca**			**Cate-goría**					0.2563	0.0763
#5			**Precio ParD**							0.1746	0.0817
#6			**Precio ParD**		**Cate-goría**					0.1644	0.0102
#7		**Marca**	**Precio ParD**							0.1290	0.0354
#8									**Rasgo**	0.1254	0.0036
#9				**Tem-porada**						0.1202	0.0052
#10				**Tem-porada**	**Cate-goría**					0.1196	0.0006
#11		**Marca**			**Cate-goría**	**Estilo**				0.1043	0.0153
#12					**Cate-goría**				**Rasgo**	0.1032	0.0011
#13			**Precio ParD**			**Estilo**				0.1011	0.0021
#14		**Marca**			**Cate-goría**	**Estilo**	**Tipo**			0.0994	0.0017
#15		**Marca**		**Tem-porada**						0.0957	0.0037
#16			**Precio ParD**			**Estilo**			**Rasgo**	0.0938	0.0019
#17					**Cate-goría**		**Tipo**			0.0910	0.0028
#18							**Tipo**			0.0910	0.0000
#19				**Tem-porada**		**Estilo**				0.0887	0.0023
#20								**Corte**		0.0811	0.0076

Hay que hacer ver que este ranking de atributos es válido para el caso concreto del estudio de la variable '*UnidadesD*' y es totalmente dependiente de los datos empleados

en dicho estudio. Los resultados podrían cambiar si se añadiera alguna otra variable al estudio, si alguna de las que se ha discretizado se volviera a discretizar con otro criterio o si la variable consecuente se cambiara por otra.

5.3.2. Clasificación del volumen de ventas

Para las siete mejores combinaciones de variables obtenidas en el epígrafe anterior, se generan los correspondientes conjuntos de reglas con el algoritmo CREA y se comentan las reglas más confiables (en verde) y también, las no confiables (en rojo). Dichas reglas comentadas son un extracto de los resultados completos obtenidos. En cada apartado se incluye una tabla que contiene todas las reglas de cada uno de los siete modelos generados, dispuestas según su grado de confiabilidad o no confiabilidad (regiones 2, 1 y 3 en verde, rojo y amarillo respectivamente). A continuación se muestran los resultados más relevantes de los siete casos estudiados, es decir las siete mejores combinaciones de variables:

Combinación #1 (ver tabla 5.9):

Regla: {Estilo} → {UnidadesD}

Tabla 5.9: Conjunto de reglas reducido para {Estilo} → {UnidadesD}

		Antec	=> Consec.
Supp(%)	Conf(%)	Estilo	UnidadesD
81.184	49.755	est_02	Num_031_050
81.184	7.462	est_02	Num_051_100
81.184	4.745	est_02	Num_101_200
81.184	2.511	est_02	Num_200_+++
2.230	38.647	est_N	Num_000_015
2.230	27.464	est_N	Num_031_050
2.230	9.890	est_N	Num_200_+++
2.230	8.533	est_N	Num_101_200
2.230	7.812	est_N	Num_051_100
2.230	7.653	est_N	Num_016_030
0.009	63.158	est_10	Num_000_015
0.009	15.789	est_10	Num_031_050
0.009	10.526	est_10	Num_051_100
0.009	7.895	est_10	Num_016_030
0.009	2.632	est_10	Num_200_+++

- El Estilo más demandado, un 81% de los pedidos, es el ‘02’
  En dichos pedidos:
  - Un 49,7% de las veces se pidieron entre 31 y 50 unidades
  - Un 7,5% entre 51 y 100 unidades
  - Un 4,7% entre 101 y 200 unidades
  - Un 2,5% de las veces se pidieron 200 o más unidades

Combinación #2 (ver tabla 5.10):

Regla: {Marca} → {UnidadesD}

- La Marca más demandada, el 80% de los pedidos, es 'MT'
  En dichos pedidos:
  - Un 49,2% de las veces se pidieron entre 31 y 50 unidades
  - Un 4,6% entre 101 y 200 unidades
  - Un 2,4% de las veces se pidieron 200 o más unidades

Tabla 5.10: Conjunto de reglas reducido para {Marca} → {UnidadesD}

		Antec	=> Consec.
Supp(%)	Conf(%)	Marca	UnidadesD
80.852	49.257	MT	Num_031_050
80.852	4.640	MT	Num_101_200
80.852	2.371	MT	Num_200_+++
1.299	79.709	EM	Num_016_030
0.990	31.623	MB	Num_016_030
0.990	21.814	MB	Num_200_+++
0.990	16.683	MB	Num_031_050
0.990	13.246	MB	Num_051_100
0.990	12.458	MB	Num_101_200
1.299	9.172	EM	Num_031_050
0.167	66.667	CH	Num_016_030
1.299	7.571	EM	Num_000_015
0.990	4.177	MB	Num_000_015
1.299	2.402	EM	Num_051_100
0.167	17.447	CH	Num_031_050
0.167	14.326	CH	Num_000_015
1.299	0.965	EM	Num_101_200
1.299	0.182	EM	Num_200_+++
0.167	1.135	CH	Num_200_+++
0.002	66.667	MV	Num_000_015
0.002	22.222	MV	Num_051_100
0.167	0.284	CH	Num_101_200
0.002	11.111	MV	Num_200_+++
0.167	0.142	CH	Num_051_100

Combinación #3 (ver tabla 5.11):

Regla: {Estilo, Marca} → {UnidadesD}

Tabla 5.11: Conjunto de reglas reducido para {Estilo, Marca} → {UnidadesD}

		Antec		=> Consec.
Supp(%)	Conf(%)	Marca	Estilo	UnidadesD
66.819	49.772	MT	est_02	Num_031_050
12.806	47.933	MT	est_01	Num_031_050
66.819	4.652	MT	est_02	Num_101_200
66.819	2.338	MT	est_02	Num_200_+++
12.806	5.744	MT	est_01	Num_051_100
12.806	4.105	MT	est_01	Num_101_200
12.806	1.561	MT	est_01	Num_200_+++
0.500	75.236	SX	est_N	Num_000_015
0.507	36.550	MM	est_N	Num_000_015
0.507	28.159	MM	est_N	Num_031_050
0.258	45.688	MB	est_01	Num_016_030
0.167	66.667	CH	est_02	Num_016_030
0.089	89.683	EM	est_01	Num_016_030
0.507	11.935	MM	est_N	Num_101_200
0.258	21.743	MB	est_01	Num_031_050
0.507	10.583	MM	est_N	Num_200_+++
0.500	9.924	SX	est_N	Num_016_030
0.507	8.811	MM	est_N	Num_051_100
0.500	8.176	SX	est_N	Num_031_050
0.258	12.202	MB	est_01	Num_200_+++
0.167	17.447	CH	est_02	Num_031_050
0.167	14.326	CH	est_02	Num_000_015
0.258	9.266	MB	est_01	Num_051_100
0.507	3.963	MM	est_N	Num_016_030
0.258	6.330	MB	est_01	Num_101_200
0.258	4.771	MB	est_01	Num_000_015
0.500	2.316	SX	est_N	Num_101_200
0.500	2.268	SX	est_N	Num_200_+++
0.500	2.079	SX	est_N	Num_051_100
0.008	65.625	MT	est_10	Num_000_015
0.089	3.968	EM	est_01	Num_031_050
0.089	3.704	EM	est_01	Num_000_015
0.167	1.135	CH	est_02	Num_200_+++
0.002	66.667	MV	est_N	Num_000_015
0.001	83.333	MB	est_N	Num_200_+++
0.008	12.500	MT	est_10	Num_031_050
0.089	1.058	EM	est_01	Num_101_200
0.089	1.058	EM	est_01	Num_051_100
0.001	50.000	MM	est_10	Num_000_015
0.008	9.375	MT	est_10	Num_051_100
0.008	9.375	MT	est_10	Num_016_030
0.001	33.333	MM	est_10	Num_031_050
0.002	22.222	MV	est_N	Num_051_100
0.089	0.529	EM	est_01	Num_200_+++
0.167	0.284	CH	est_02	Num_101_200
0.001	16.667	MM	est_10	Num_051_100
0.001	16.667	MB	est_N	Num_101_200
0.002	11.111	MV	est_N	Num_200_+++
0.008	3.125	MT	est_10	Num_200_+++
0.167	0.142	CH	est_02	Num_051_100

- La combinación Marca ‘MT’ y Estilo ‘02’ es la más demandada con un 66,8%
  En dichos pedidos:
  - Un 49,7% de las veces se pidieron entre 31 y 50 unidades
  - Un 4,6% entre 101 y 200 unidades
  - Un 2,3% de las veces se pidieron 200 o más unidades

- La combinación Marca ‘MT’ y Estilo ‘01’ se demandada con un 12,8%
  En dichos pedidos:
  - Un 47,9% de las veces se pidieron entre 31 y 50 unidades
  - Un 5,7% entre 51 y 100 unidades
  - Un 4,1% entre 101 y 200 unidades
  - Un 1.5% de las veces se pidieron 200 o más unidades

Combinación #4 (ver tabla 5.12):

Regla: {Marca, Categoría} → {UnidadesD}

- La combinación Marca ‘MT’ y Categoría ‘Z’ se demanda un 67,9%
  En dichos pedidos:
  - Un 58,2% de las veces se pidieron entre 31 y 50 unidades
  - Un 5,4% 15 o menos unidades
  - Un 5,4% entre 101 y 200 unidades
  - Un 2,7% 200 o más unidades

- La combinación Marca ‘MT’ y Categoría ‘C’ se demanda un 12,8%
  Entre dichos pedidos:
  - Un 90,6% de las veces se piden 15 o menos Unidades
  - Un 5,8% entre 16 y 30 unidades
  - Un 1,8% entre 31 y 50 unidades
  - Un 1,2% entre 51 y 100 unidades
  - Un 0,4% entre 101 y 200 unidades
  - Un 0,2% 200 o más unidades

Tabla 5.12: Conjunto de reglas reducido para {Marca, Categoría} → {UnidadesD}

		Antec		=> Consec.
Supp(%)	Conf(%)	Marca	Categoria	UnidadesD
67.973	58.248	MT	Z	Num_031_050
12.870	90.597	MT	C	Num_000_015
67.973	5.448	MT	Z	Num_000_015
67.973	5.436	MT	Z	Num_101_200
67.973	2.780	MT	Z	Num_200_+++
12.870	5.787	MT	C	Num_016_030
12.870	1.798	MT	C	Num_031_050
12.870	1.172	MT	C	Num_051_100
12.870	0.441	MT	C	Num_101_200
12.870	0.206	MT	C	Num_200_+++
0.167	66.667	CH	Z	Num_016_030
0.167	17.447	CH	Z	Num_031_050
0.167	14.326	CH	Z	Num_000_015
0.008	60.000	MT	P	Num_000_015
0.004	58.824	SX	C	Num_016_030
0.167	1.135	CH	Z	Num_200_+++
0.004	41.176	SX	C	Num_000_015
0.002	66.667	MV	Z	Num_000_015
0.008	11.429	MT	P	Num_031_050
0.001	50.000	MM	P	Num_000_015
0.008	8.571	MT	P	Num_016_030
0.008	8.571	MT	P	Num_051_100
0.008	8.571	MT	P	Num_200_+++
0.001	33.333	MM	P	Num_031_050
0.002	22.222	MV	Z	Num_051_100
0.167	0.284	CH	Z	Num_101_200
0.001	16.667	MM	P	Num_051_100
0.002	11.111	MV	Z	Num_200_+++
0.008	2.857	MT	P	Num_101_200
0.167	0.142	CH	Z	Num_051_100

Combinación #5 (ver tabla 5.13):

Regla: {PrecioParD} → {UnidadesD}

- El PrecioParD más repetido, un 44,6% de las veces, es entre 25 y 50
  Entre dichos pedidos:
  - Un 49,2% de las veces se pidieron entre 31 y 50 unidades
  - Un 4,3% entre 101 y 200 unidades
  - Un 2,0% más de 200 unidades

- Un 29,2% de las veces se da el PrecioParD entre 50 y 75
  Entre dichos pedidos:
  - Un 44,4% de las veces se pidieron entre 31 y 50 unidades
  - Un 3,1% entre 101 y 200 unidades
  - Un 1,3% más de 200 unidades

Tabla 5.13: Conjunto de reglas reducido para {PrecioParD} → {UnidadesD}

		Antec	=> Consec.
Supp(%)	Conf(%)	PrecioParD	UnidadesD
44.637	49.199	pre_025_050	Num_031_050
29.179	44.432	pre_050_075	Num_031_050
44.637	4.327	pre_025_050	Num_101_200
44.637	2.019	pre_025_050	Num_200_+++
29.179	3.072	pre_050_075	Num_101_200
29.179	1.357	pre_050_075	Num_200_+++
5.190	52.589	pre_100_+++	Num_016_030
3.438	53.290	pre_075_100	Num_016_030
5.190	31.996	pre_100_+++	Num_031_050
3.438	29.969	pre_075_100	Num_031_050
5.190	5.756	pre_100_+++	Num_051_100
3.438	8.443	pre_075_100	Num_000_015
5.190	3.971	pre_100_+++	Num_101_200
3.438	4.758	pre_075_100	Num_051_100
5.190	2.869	pre_100_+++	Num_000_015
5.190	2.819	pre_100_+++	Num_200_+++
3.438	2.702	pre_075_100	Num_101_200
3.438	0.839	pre_075_100	Num_200_+++

Combinación #6 (ver tabla 5.14):

Regla: {Categoría, PrecioParD} → {UnidadesD}

- La combinación Categoría ‘Z’ y PrecioParD ‘25-50’ se da un 37,4%
  Entre dichos pedidos:
  - Un 58,3% de las veces se pidieron entre 31 y 50 unidades
  - Un 2,4% se pidieron más de 200 unidades

- La combinación Categoría ‘Z’ y PrecioParD ‘00-25’ se da un 17,1%
  Entre dichos pedidos:
  - Un 64,8% de las veces se pidieron entre 31 y 50 unidades
  - Un 3,3% se pidieron entre 16 y 30 unidades

Tabla 5.14: Conjunto de reglas reducido para {Categoría, PrecioParD} → {UnidadesD}

		Antec		=> Consec.
Supp(%)	Conf(%)	Categoria	PrecioParD	UnidadesD
37.405	58.356	Z	pre_025_050	Num_031_050
17.051	64.782	Z	pre_000_025	Num_031_050
37.405	2.382	Z	pre_025_050	Num_200_+++
24.032	3.685	Z	pre_050_075	Num_101_200
17.051	3.303	Z	pre_000_025	Num_016_030
24.032	1.645	Z	pre_050_075	Num_200_+++
0.495	78.950	C	pre_000_025	Num_000_015
0.495	8.067	C	pre_000_025	Num_016_030
0.495	3.532	C	pre_000_025	Num_051_100
0.495	3.389	C	pre_000_025	Num_031_050
0.495	3.198	C	pre_000_025	Num_200_+++
0.495	2.864	C	pre_000_025	Num_101_200
0.010	58.537	P	pre_000_025	Num_000_015
0.010	14.634	P	pre_000_025	Num_031_050
0.010	9.756	P	pre_000_025	Num_051_100
0.010	7.317	P	pre_000_025	Num_200_+++
0.010	7.317	P	pre_000_025	Num_016_030
0.010	2.439	P	pre_000_025	Num_101_200

Combinación #7 (ver tabla 5.15):

Regla: {Marca, PrecioParD} → {UnidadesD}

Tabla 5.15: Conjunto de reglas reducido para {Marca, PrecioParD} → {UnidadesD}

		Antec		=> Consec.
Supp(%)	Conf(%)	Marca	PrecioParD	UnidadesD
16.330	64.755	MT	pre_000_025	Num_031_050
16.330	5.894	MT	pre_000_025	Num_200_+++
16.330	5.616	MT	pre_000_025	Num_000_015
16.330	3.231	MT	pre_000_025	Num_016_030
0.042	34.091	MB	pre_000_025	Num_016_030
0.042	32.955	MB	pre_000_025	Num_200_+++
0.032	39.259	SX	pre_000_025	Num_000_015
0.042	21.591	MB	pre_000_025	Num_101_200
0.032	25.185	SX	pre_000_025	Num_031_050
0.032	21.481	SX	pre_000_025	Num_016_030
0.042	7.955	MB	pre_000_025	Num_051_100
0.032	6.667	SX	pre_000_025	Num_051_100
0.002	80.000	CH	pre_000_025	Num_200_+++
0.032	5.185	SX	pre_000_025	Num_101_200
0.002	66.667	MV	pre_000_025	Num_000_015
0.042	3.409	MB	pre_000_025	Num_031_050
0.032	2.222	SX	pre_000_025	Num_200_+++
0.002	22.222	MV	pre_000_025	Num_051_100
0.002	20.000	CH	pre_000_025	Num_101_200
0.002	11.111	MV	pre_000_025	Num_200_+++

- La combinación Marca ‘MT’ y PrecioParD ‘25-50’ se da un 35,8%
  Entre dichos pedidos:
  - Un 47,4% de las veces se pidieron entre 31 y 50 unidades
  - Un 4,1% entre 101 y 200 unidades
  - Un 1,7% más de 200 unidades

- La combinación Marca ‘MT’ y PrecioParD ‘50-75’ se da un 25,3%
  Entre dichos pedidos:
  - Un 44% de las veces se pidieron entre 31 y 50 unidades
  - Un 5,0% entre 51 y 100 unidades
  - Un 3,0% entre 101 y 200 unidades
  - Un 1,2% más de 200 unidades

- La combinación Marca ‘MT’ y PrecioParD ‘00-25’ se da un 35,8%
  Entre dichos pedidos:
  - Un 47,4% de las veces se pidieron entre 31 y 50 unidades
  - Un 4,1% entre 101 y 200 unidades
  - Un 1,7% más de 200 unidades

5.3.3. Estudio de la complejidad computacional

Con el objeto de completar el proceso de análisis de los datos desde el punto de vista del rendimiento del algoritmo CERA, se ha realizado un tercer estudio que ha consistido en obtener reglas de clasificación sobre la variable ‘*UnidadesD*’ empleando todos los campos disponibles para el consecuente, repitiendo el proceso eliminando uno, dos, ... y hasta ocho de dichos campos, tomando en cada caso el tiempo empleado por el algoritmo en realizar el cálculo. En la tabla 5.16 se muestran los resultados obtenidos de la medición de dichos tiempos de procesamiento para las nueve combinaciones de variables consideradas en este estudio.

Para tratar de ubicar los tramos de la función de complejidad en este estudio, como ya se comentó anteriormente, se debe asimilar la potencia ‘V^C’ al producto del número de valores de las columnas que intervienen en cada ejecución. Es decir, el número de valores de las variables antecedentes que corresponda en cada caso, junto con el número de valores de la variable consecuente ‘*UnidadesD*’. Una vez calculado el valor ‘V^C’ se puede obtener un valor aproximado de ‘V’ calculando la raíz ‘C-ésima’ sobre ‘V^C’ y a partir de dicho valor de ‘V’ se puede determinar, también de forma aproximada, el valor de la potencia ‘V^{C+2}’, necesaria para localizar los tramos de la función de complejidad. Recuérdese que en este estudio el valor ‘N’ es de 423085.

Tabla 5.16: Tiempos de cómputo para los 7 antecedentes considerados

	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	Tipo	Corte	Rasgo	Tiempo (clocks)
Ant-1	Sem Ped	---	---	---	---	---	---	---	---	200.1
Ant-2	Sem Ped	Marca	---	---	---	---	---	---	---	250.4
Ant-3	Sem Ped	Marca	Precio ParD	---	---	---	---	---	---	444.1
Ant-4	Sem Ped	Marca	Precio ParD	Tem-porada	---	---	---	---	---	607.8
Ant-5	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	---	---	---	---	638.6
Ant-6	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	---	---	---	986.5
Ant-7	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	Tipo	---	---	1365.7
Ant-8	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	Tipo	Corte	---	1548.0
Ant-9	Sem Ped	Marca	Precio ParD	Tem-porada	Cate-goría	Estilo	Tipo	Corte	Rasgo	1865.8

Tabla 5.17: Tramos para cada antecedente considerado

Antecedente	C	V^C	~V	$\sim V^{C+2}$	Tramo
Ant-1	2	312	17.66	97244.00	T3
Ant-2	3	1872	12.32	284343.35	T3
Ant-3	4	9360	9.84	905552.79	T2
Ant-4	5	131040	10.56	14600422.99	T2
Ant-5	6	393120	8.56	28798300.04	T2
Ant-6	7	1572480	7.68	92691097.60	T1
Ant-7	8	22014720	8.28	1507965680.69	T1
Ant-8	9	550368000	9.36	48197059969.55	T1
Ant-9	10	15960672000	10.48	1752513641791.11	T1

Gráficamente, para estas nueve combinaciones de posibles antecedentes, se obtiene la evolución temporal que muestra la figura 5.31. Como se puede ver en la gráfica, la evolución del tiempo de ejecución del algoritmo presenta los tres tramos compatibles con la función teórica de la complejidad computacional, aunque se observa que dichos tramos se encuentran ligeramente desplazados a la derecha con respecto al cálculo teórico. Como ya se comentó en el epígrafe 5.2.7, este desplazamiento se debe al hecho de que la función de la complejidad teórica no es un reflejo exacto de la experiencia computacional (sino más bien, una cota superior). Además, en este caso concreto se han realizado una serie de aproximaciones para el cálculo de los valores 'V' y 'V^{C+2}' que pueden estar acentuando un poco más la diferencia entre el estudio teórico y los tiempos empíricamente obtenidos.

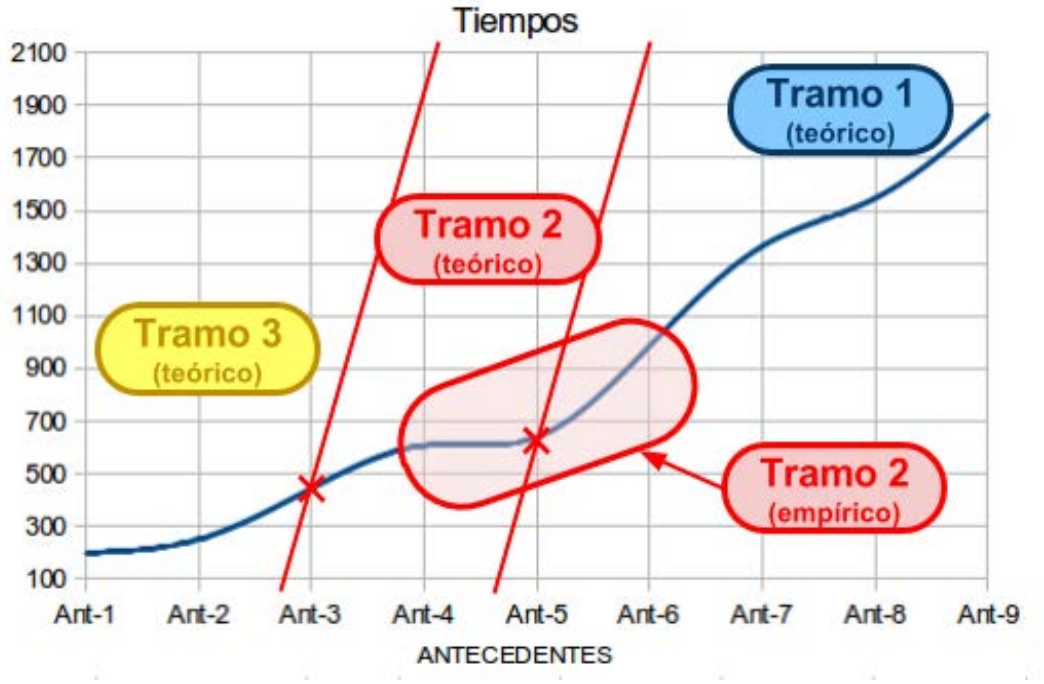


Figura 5.31: Tiempo para los 9 antecedentes considerados

5.3.4. Conclusiones y sugerencias

Más allá de los resultados computacionales, la incorporación de datos empresariales reales en esta Tesis, tiene como objetivo demostrar la validez e idoneidad del método predictivo propuesto, dentro del ámbito estratégico empresarial. Así pues, y a la vista de los resultados contrastados con los expertos de la empresa, se concluye:

- No hay reglas importantes del negocio que no queden recogidas por el sistema de reglas resultante.

- El sistema de reglas obtenido clasifica de manera coherente las principales reglas del negocio, no mostrando conflictos con el *know-how* empresarial (reforzando el conocimiento previo) pero descubriendo nuevos patrones de venta que hasta la fecha habían recibido menor atención.

- La obtención de un ranking ponderado de las combinaciones de variables más relevantes de cara a modelar la variable objetivo '*UnidadesD*' es valorada muy positivamente desde el punto de vista estratégico pues permite semi-automatizar la explotación de cuadros de mando, en función de métricas exactas de correlación, optimizando el proceso de selección de características, que normalmente llevan a cabo los analistas de manera completamente manual, basándose exclusivamente en experiencias previas y en la rutina de manejo.

- Se perciben grandes potencialidades derivadas de la posible aplicación del método a nuevos retos dentro de la organización, como por ejemplo: la identificación automática de clientes generadores de tendencia de compra, o el descubrimiento de patrones de navegación en portal de comercio electrónico, entre otras.

No obstante, a pesar de la coherencia de los resultados obtenidos se hace ver a la empresa las siguientes apreciaciones, que ayudaran a mejorar los siguientes análisis:

- Los resultados obtenidos presentan importantes sesgos por la naturaleza de algunos de los atributos de que se disponía (‘*Estilo*’, ‘*Marca*’, ‘*Categoría*’ y ‘*Rasgo*’), así como de la variable objetivo ‘*UnidadesD*’. Por ello, con demasiada frecuencia se obtienen combinaciones de valores donde el ‘*Estilo*’ toma el valor ‘02’, la ‘*Marca*’ el valor ‘MT’, o la consecuencia inferida por la regla es que las unidades pedidas son ‘31 - 50’.

- Este problema podría resolverse en parte realizando estudios más refinados en los que en la fase de preprocesamiento se debería preseleccionar subconjuntos de registros que cumplan una determinada característica y realizar estudios independientes para un subgrupo de datos. Esta solución proporcionaría resultados independientes, por ejemplo, para cada estilo o para cada marca o para cada combinación de estilo y marca. El problema estaría en que mientras los estudios así realizados para subconjuntos de datos en los que una determinada variable (o conjunto de ellas) tomara muchas veces un mismo valor tendrían un alto soporte, en los casos en los que algún valor se repitiera pocas veces los resultados, siendo técnicamente correctos, tendrían un soporte menor.

- Merece una especial atención el campo ‘*SemPed*’ (Semana Pedido) extraído de la discretización de la fecha de los pedidos realizados. Debido a la dispersión de sus valores, esta variable no es identificada como importante a efectos de clasificar las unidades de producto pedidas en cada pedido, sin embargo, es obvio que a efectos predictivos es un valor muy relevante. Para este caso, se propone como sugerencia, realizar estudios específicos por semanas o grupos de semanas (quincenas o meses) y hacer comparativas entre semanas para buscar patrones que permitan inferir niveles de pedidos futuros en función de otros previos.

- Igualmente, sería posible plantear estudios en los que agrupar las ventas por clientes y hacer análisis específicos para algunos de ellos (por ejemplo, los que realizan más compras de un determinado producto o de una determinada marca). Estos resultados pueden compararse con los obtenidos en otros estudios más generalistas y así detectar si alguno de los clientes estuviera marcando una tendencia de compra con respecto al total de los pedidos realizados.

5.4. Comparativa con métodos clásicos

Cualquier sistema generador de reglas se mide bajo dos métricas fundamentales: la precisión y el tiempo de ejecución.

5.4.1. Precisión

Dado que el algoritmo CREA utiliza los mismos criterios de pre-poda que el método ID3, es lógico que los umbrales de precisión de las reglas obtenidas sean los mismos para ambos algoritmos. La precisión depende del conjunto de datos de entrada. Sobre los conjuntos de datos utilizados en esta Tesis, dichos umbrales de precisión varían entre el 72% y el 91%.

Los criterios de post-poda empleados por CREA son los del algoritmo RBS. Tal y como se describe en (Rabasa 2009), las únicas reglas eliminadas a posteriori son las reglas con soporte y confianzas muy poco significativos que en cualquier caso no van a afectar al modelo final de clasificación.

5.4.2. Tiempos de Ejecución

La mayor parte del esfuerzo en la fase de implementación del método presentado en esta Tesis se invirtió en el diseño óptimo del algoritmo y sus estructuras de datos, de manera que, gracias a una gestión eficaz de la memoria y al diseño de unas estructuras de datos que permiten reducir el número de iteraciones, se consigue disminuir significativamente los tiempos de ejecución del ID3, método de clasificación de referencia, así como los del método RBS.

De forma resumida, en las tablas 5.18, 5.19 y 5.20 se muestran respectivamente los tiempos de ejecución del algoritmo CREA (implementado en C++), el algoritmo RBS (precursor del presentado en esta Tesis, implementado también en C++) y el algoritmo ID3 proporcionado por la aplicación WEKA (software genérico de Minería de Datos implementado en JAVA).

Tabla 5.18: Resumen de tiempos, algoritmo CREA

C	$2x10^5$	$4x10^5$	$6x10^5$	$8x10^5$	$10x10^5$	$12x10^5$	$14x10^5$	$16x10^5$	$18x10^5$	$20x10^5$
2	53.3	106.1	158.8	211.5	264.4	316.7	370.1	423.6	476.4	530.2
4	314.6	524.8	729.9	932.1	1154.0	1343.9	1557.0	1779.7	1969.1	2280.0
8	2458.5	4924.8	7004.9	9972.4	11502.4	14574.7	17518.6	20040.6	19978.2	22616.5
16	4961.1	10187.1	15828.7	21837.8	27841.5	34520.0	41078.8	46757.8	53658.7	58769.3
32	10438.7	22087.5	33335.7	48982.0	60673.6	73954.2	91467.0	109045.3	116514.4	131214.1

Tabla 5.19: Resumen de tiempos, algoritmo RBS

C	$2x10^5$	$4x10^5$	$6x10^5$	$8x10^5$	$10x10^5$	$12x10^5$	$14x10^5$	$16x10^5$	$18x10^5$	$20x10^5$
2	824.7	1979.4	2969.1	5278.4	8247.4	11876.3	16164.9	21113.4	26721.6	32989.7
4	1649.5	3958.8	5938.1	10556.7	16494.8	23752.6	32329.9	42226.8	53443.3	65979.4
8	3299.0	7917.5	11876.3	21113.4	32989.7	47505.2	64659.8	84453.6	106886.6	131958.8
16	6597.9	15835.1	23752.6	42226.8	65979.4	95010.3	129319.6	168907.2	213773.2	263917.5
32	13195.9	31670.1	47505.2	84453.6	131958.8	190020.6	258639.2	337814.4	427546.4	527835.1

Tabla 5.20: Resumen de tiempos, algoritmo ID3 (WEKA)

C	$2x10^5$	$4x10^5$	$6x10^5$	$8x10^5$	$10x10^5$	$12x10^5$	$14x10^5$	$16x10^5$	$18x10^5$	$20x10^5$
2	110.0	250.0	300.0	340.0	410.0	570.0	690.0	780.0	890.0	1020.0
4	290.0	730.0	970.0	1480.0	2270.0	2990.0	3420.0	4110.0	5630.0	7190.0
8	3830.0	7700.0	X	X	X	X	X	X	X	X
16	8110.0	18770.0	30420.0	X	X	X	X	X	X	X
32	16870.0	41740.0	X	X	X	X	X	X	X	X

Queda patente que el algoritmo CREA mejora los tiempos de ejecución de RBS y de ID3-WEKA. En las figuras 5.32 y 5.33 se representan de manera comparativa los tiempos de ejecución de estos tres métodos para los conjuntos de datos correspondientes a los valores de 'N': '$2x10^5$' y '$4x10^5$' respectivamente.

Estas ejecuciones se han realizado sobre los conjuntos de datos simulados. En cada caso, los valores representados se corresponden con el tiempo promedio de ejecución para los 9 valores de 'V' considerados en dichos conjuntos de datos. Debe observarse que la implementación del método RBS a que se está haciendo referencia consiste en un mecanismo de generación de reglas basado en un conteo de frecuencias a partir de un fichero de datos. Realizado dicho conteo se dispone de un conjunto de reglas, cada una de las cuales caracterizada con sus correspondientes valores de soporte y confianza. Tras la generación de dichas reglas se aplica el algoritmo RBS propiamente dicho para reducir el citado conjunto.

Sobre la experiencia computacional realizada con la aplicación WEKA cabe decir que no ha sido posible completarla ya que esta aplicación no ha podido manejar algunos conjuntos de datos cuando éstos eran superiores a un cierto tamaño (p.e: para '$N=6x10^5$' y 'C=8' → 24.6 MB; para '$N=6x10^5$' y 'C=32' → 96.6 MB; para '$N=8x10^5$' y 'C=8' → 32.8 MB; etc). En algunas ocasiones se ha observado que al ejecutar el algoritmo ID3 la aplicación no ha respondido y, por tanto, no proporcionó una salida válida, en otras ocasiones, directamente, la aplicación no ha sido capaz de cargar en memoria los datos suministrados (antes de ejecutar ID3). Debe tenerse en cuenta también que esta aplicación (WEKA) proporciona los tiempos de ejecución en segundos, con una precisión de centésimas de segundo (no de milisegundos como en los métodos CREA y RBS).

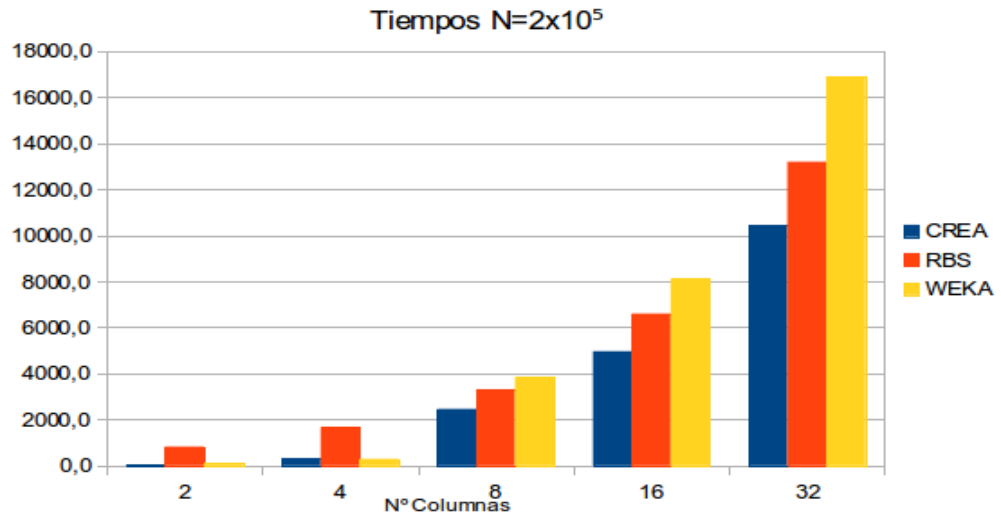


Figura 5.32: Comparativa tiempos CREA vs RBS vs ID3-WEKA para N=2x10^5

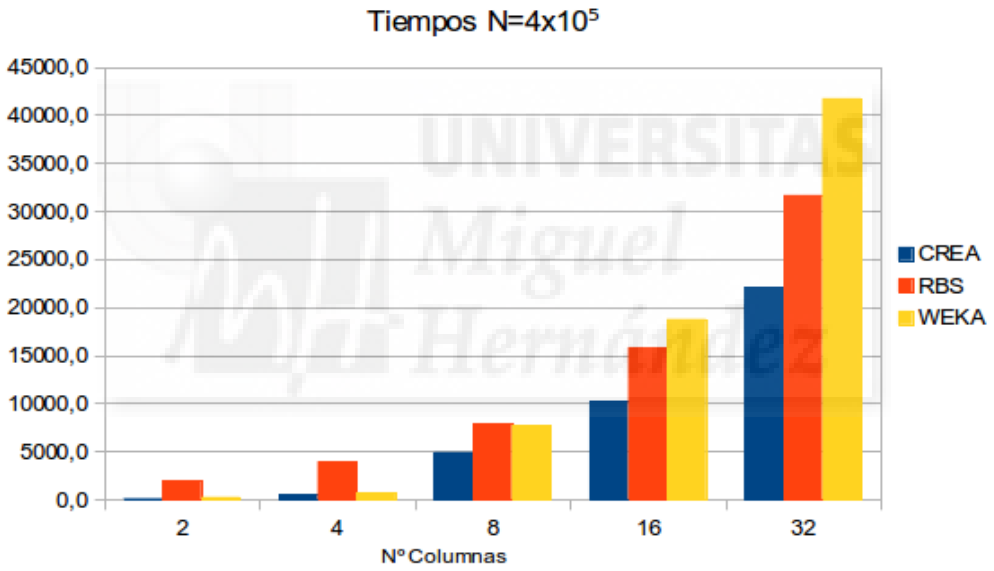


Figura 5.33: Comparativa tiempos CREA vs RBS vs ID3-WEKA para N=4x10^5

Finalmente, queda hacer constar que todas las pruebas realizadas para la observación de los tiempos de procesamiento se han realizado en el mismo equipo informático. Se trata de un PC de sobremesa con procesador Intel Pentium 4 a 2 Ghz y 2 MBytes de memoria RAM, con Windows XP SP2 como sistema operativo.

UNIVERSITAS
Miguel
Hernández

Capítulo 6
Conclusiones

6.1. Validez y limitaciones de CREA

Al igual que ya ocurría con su precursor (el método RBS), el método CREA ordena por significancia los sistemas de reglas, pero incorpora una ventaja sustancial que lo hace especialmente válido para el contexto de negocio, y es que, en general, el algoritmo CREA ofrece muy buenos resultados desde el prisma de la complejidad computacional. Gracias a las estructuras de datos y a la gestión de memoria que se ha implementado, se consiguen reducciones de tiempo considerables frente a métodos de clasificación clásicos.

Por otra parte, aplicar la nueva métrica 'WACI' al problema de la extracción automática de características más relevantes ha proporcionado resultados satisfactorios en el ámbito empresarial en el que se ha realizado la prueba (datos comerciales de la empresa MUSTANG). Al igual que la métrica original, 'ACI', en el trabajo de Zaragozí et al. (2011), esta modificación de la citada métrica ha permitido identificar qué atributos, de entre los seleccionados inicialmente para el proceso, son los más relevantes para clasificar los ejemplos del conjunto de datos para una determinada variable de clase ('*UnidadesD*'), por lo que se puede concluir que la métrica 'WACI' ha significado un potente método para la identificación de los atributos más importantes, desarrollándose un método automático que permite comparar los conjuntos de características y

determinar cuáles son estadísticamente más influyentes de cara a la predicción de un determinado valor objetivo.

Más concretamente, sobre el análisis de los datos empresariales de venta de calzado a minoristas, que ha sido validado por personal cualificado de la empresa, el conocimiento aportado se muestra de especial interés para los responsables de la toma de decisiones. Esta circunstancia, tal y como ya se apuntaba en el epígrafe 5.3.4, se debe principalmente a dos circunstancias:

- Las reglas de clasificación más importantes son coherentes con el *know-how* de la empresa, pero también aportan nuevo conocimiento estratégico sobre cómo es el comportamiento de la venta a minoristas.

- La extracción automática de las variables más significativas y la ponderación de su relevancia frente al volumen de ventas se muestra como un gran valor añadido que permite replantear la explotación de los cuadros de mando que se vienen empleando, y así, poder sacarles un mayor partido.

Como contrapartida, CREA sólo es aplicable en problemas de clasificación donde la variable objeto de estudio (el consecuente de la regla), es una variable de clase, es decir, es de tipo nominal o discreta. Además, el resto de variables que intervienen en el problema (atributos antecedentes) también deben ser de tipo nominal. Por ello, en problemas reales en el ámbito empresarial, es frecuente recurrir a la discretización de variables numéricas, en la etapa de preprocesamiento, antes de aplicar el algoritmo. Dicho preprocesamiento se realiza habitualmente por segmentación manual del intervalo según los criterios que fijan los expertos en cada caso. Si el usuario no indica criterio alguno para la discretización, se lleva a cabo un proceso semi-automático para dicha tarea en función de las distribuciones de los valores en cada atributo.

6.2. Propuesta de una métrica de calidad de los datos

Cualquier proceso de Minería de Datos es totalmente dependiente de la calidad del conjunto de ejemplos de entrada ‘E’. Si dicho conjunto de datos contiene ruido o está sesgado por algún valor en alguno de sus atributos, el modelo resultante heredará ese problema. Para estos casos resultaría interesante disponer de alguna métrica que permita determinar dicha calidad. A raíz de los experimentos realizados, se plantea, como futura línea de investigación, la definición de una métrica ‘Q_E’, de la calidad de una muestra de ejemplos ‘E’, en función del número de reglas del modelo generado.

De acuerdo con los principios de la Teoría Matemática de la Comunicación (Shannon 1948) la información es más relevante en la medida que sea más discriminatoria y exclusiva. Por ello, la calidad de la muestra debe ser inversamente proporcional al

conjunto de reglas significativas, y directamente proporcional al tamaño de la muestra o a la combinatoria de reglas a proporcionar. Así, la calidad de la muestra, 'Q_E', debería ser proporcional al cociente de la expresión 6.1.

$$Q_E \approx \frac{\text{Min}(N, V^C)}{|Rr|} \qquad \text{(Expresión 6.1)}$$

Siendo:

- '*N*': El número de ejemplos del conjunto de datos 'E'.
- 'V^C': Número de posibles combinaciones de todos los valores de los atributos del conjunto de datos (si los 'C' atributos tienen el mismo número de valores, el valor coincide con la exponencial 'V^C', si no, se aplica la expresión 5.3).
- '*Rr*': Conjunto de reglas reducido generado por CREA, es decir, las reglas significativas que quedan tras eliminar las de la región 'REG-0'.

Tabla 6.1: Cálculo aproximado de 'Q_E' para los subconjuntos de datos empresariales

	Conjunto 'E'	**V^C**	**Rr**	**Q_E**
#1	V6, V10	24	15	1.60
#2	V2, V10	42	24	1.75
#3	V2, V6, V10	168	50	3.36
#4	V2, V5, V10	126	30	4.20
#5	V3, V10	30	18	1.66
#6	V3, V5, V10	90	18	5.00
#7	V2, V3, V10	210	89	2.35
#8	V9, V10	174	78	2.23
#9	V4, V10	84	53	1.58
#10	V4, V5, V10	252	72	3.50
#11	V2, V5, V6, V10	504	14	36.00
#12	V5, V10	18	7	2.57
#13	V3, V6, V10	120	9	13.33
#14	V2, V5, V6, V7, V10	7056	3	2.35
#15	V2, V4, V10	588	116	5.06
#16	V3, V6, V9, V10	3480	28	124.28
#17	V5, V7, V10	252	10	25.20
#18	V7, V10	84	49	1.17
#19	V4, V6, V10	336	66	5.09
#20	V8, V10	150	85	1.76
Ant-1	V1, V10	312	176	1.77
Ant-2	V1, V2, V10	2184	645	3.38
Ant-3	V1, V2, V3, V10	10920	1593	6.85
Ant-4	V1, V2, V3, V4, V10	152880	2389	6399
Ant-5	V1, V2, V3, V4, V5, V10	458640	2113	200.22
Ant-6	V1, V2, V3, V4, V5, V6, V10	1834560	1302	324.95
Ant-7	V1, V2, V3, V4, V5, V6, V7, V10	25683840	3039	139.21
Ant-8	V1, V2, V3, V4, V5, V6, V7, V8, V10	642096000	3500	120.88
Ant-9	V1, V2, V3, V4, V5, V6, V7, V8, V9, V10	18620784000	5116	82.69

La tabla 6.1 muestra todos los valores del cociente de la expresión 6.1 (valor proporcional a la métrica 'Q_E' que se propone investigar) para los diferentes conjuntos de ejemplos considerados en el análisis de los datos empresariales (ver tablas 5.8 y 5.16). Nótese que en todos los casos el valor 'N' es de 423085 registros. Las etiquetas 'Vi' de la tabla 6.1 se corresponden con los atributos del conjunto de datos según la siguiente descripción:

- V1 → Sem-Ped
- V2 → Marca
- V3 → PrecioParD
- V4 → Temporada
- V5 → Categoría
- V6 → Estilo
- V7 → Tipo
- V8 → Corte
- V9 → Rasgo
- V10 → UnidadesD

6.3. Consecución de objetivos

Tal y como se indicó en el epígrafe 2.2, esta Tesis se plantea unos objetivos fundamentales, y es en este capítulo de conclusiones donde corresponde recapitular sobre el grado de consecución de los mismos.

Como primer objetivo se planteaba **plasmar un estado del arte con clasificación bibliográfica formal del papel del Data Mining en problemas predictivos de Inteligencia de Negocios**. El capítulo 1 presenta dicho estudio del estado del arte, desde lo más general a lo más concreto, introduciendo los conceptos fundamentales de Minería de Datos y de Inteligencia de Negocios hasta plantear qué técnicas son las más empleadas habitualmente, para resolver qué tipos de problemas. Dicho capítulo concluye con una tabla resumen (tabla 1.2) que posiciona los principales estudios existentes en la materia, agrupados por técnicas y alcance.

El segundo de los objetivos planteados era el **desarrollo un algoritmo de generación y reducción de reglas de clasificación capaz de generar sistemas de reglas precisas y ordenadas en tiempos mínimos**. Dicho algoritmo es el CREA cuya complejidad temporal, tal y como se ha demostrado teórica y empíricamente, es sustancialmente mejor que la de los algoritmos de clasificación de su misma tipología (ID3 y RBS).

En tercer lugar, se proponía **realizar un estudio formal de la complejidad computacional del algoritmo propuesto y verificar que el resultado del mismo es**

coherente con las pruebas empíricas realizadas. La primera parte de este objetivo queda cubierta con el detallado estudio teórico de la complejidad computacional desarrollado en el epígrafe 3.4. En el capítulo 5 se han desarrollado numerosas pruebas empíricas sobre un gran número de conjuntos de datos (simulados y reales), y ha quedado patente que los resultados de dichas pruebas empíricas son coherentes con el resultado del estudio teórico del coste computacional de CREA.

El cuarto objetivo era **introducir un sistema de obtención de características más relevantes de entre las disponibles (*'feature selection'*) utilizando la métrica 'ACI' proporcionada por el algoritmo RBS**. Sobre este punto, se ha planteado una variante de la métrica 'ACI', que se ha denominado 'WACI', que ha proporcionado resultados válidos en la extracción de características más relevantes de entre las disponibles en el conjunto de datos reales de la empresa MUSTANG.

Por último, en los objetivos propuestos se planteaba comprobar si se cumple o no la siguiente hipótesis: **Dados dos conjuntos de datos con igual número de ejemplos, si se cumple que la potencia 'V^C' coincide en ambos casos, los tiempos de ejecución del algoritmo CREA para dichas muestras coinciden o son muy parecidos**. Las pruebas realizadas en el apartado 5.2.7 contradicen la hipótesis, por lo que finalmente debe rechazarse.

6.4. Líneas futuras de investigación

El presente trabajo se ha querido englobar dentro del campo de las técnicas de análisis de datos aplicables a la Inteligencia de Negocios y a la toma de decisiones estratégicas por parte de ejecutivos o directivos de empresas. Si bien el ámbito empresarial no es el único en el que se pueden aplicar estas técnicas, sí es cierto que se ha puesto el foco en la problemática de la toma de decisiones en el citado entorno empresarial. Los responsables de la toma de decisiones, cada vez más, tienen que lidiar con grandes volúmenes de datos (Big Data) para obtener una información fiable que les permita tomar mejores decisiones y, con frecuencia, necesitan disponer de dicha información con premura. Consideramos que la principal aportación de esta Tesis es el algoritmo CREA que aúna en un mismo procedimiento un método para generar reglas de clasificación junto con un mecanismo de reducción y ordenación de las mismas. Este algoritmo permite generar de forma rápida modelos reducidos de reglas de clasificación más fácilmente interpretables por un ejecutivo. Sin embargo este sistema tal cual está diseñado solamente permite resolver un tipo de problemas (clasificación) a los que se pueden enfrentar los directivos de empresa, aunque creemos que con él es posible sentar las bases para futuras investigaciones y desarrollos que permitirán satisfacer más necesidades de este colectivo (sin menoscabo de otros ámbitos o colectivos).

6.4.1. Mejora de la selección de características más relevantes

La nueva métrica 'WACI' ha resultado muy útil para la extracción de características relevantes. Como ya se describió, el mecanismo de extracción de estas características consiste en hacer sucesivas ejecuciones del algoritmo CREA cambiando los atributos consecuentes de las reglas que se van a generar. Aquella combinación de atributos que proporciona un mayor valor del 'WACI' será la que tenga una mayor relevancia para clasificar el consecuente elegido. En el epígrafe 5.3.1 se describe más detalladamente cómo se ha aplicado esta técnica a los datos de la empresa MUSTANG para identificar las combinaciones de atributos que mejor sirven para calificar la variable '*UnidadesD*'. En este caso concreto ha sido factible implementar una solución de fuerza bruta ya que las variables que podían formar parte del antecedente eran 9, por tanto había 511 combinaciones que probar (2^9-1). Esta ejecución, gracias a la eficiencia del algoritmo CREA puede realizarse en unos pocos minutos, que es un tiempo más que asumible. Sin embargo, independientemente de la eficiencia de CREA, verificar todas las posibles combinaciones de un conjunto de atributos es un problema intrínsecamente exponencial. Para 'C' atributos, las combinaciones serían '2^C-1', por lo que una solución de fuerza bruta no será válida para números de atributos un poco mayores. Por ejemplo, asumiendo que cada ejecución de CREA tarda un segundo (en la práctica podrá ser más o menos dependiendo del conjunto de entrada y de la potencia de la máquina que ejecute el proceso), para 9 atributos antecedentes el proceso tardaría:

$$C=9 \rightarrow 29\text{-}1 = 511 \text{ segundos} \rightarrow 8{,}516 \text{ minutos} \qquad \text{(Expresión 6.2)}$$

En cambio, para 15 atributos se tiene que:

$$C=15 \rightarrow 215\text{-}1 = 32767 \text{ segundos} \rightarrow 9{,}101 \text{ horas} \qquad \text{(Expresión 6.3)}$$

Si hubiera que considerar 30 atributos el tiempo de ejecución sería:

$$C=30 \rightarrow 2^{30}\text{-}1 = 1073741823 \text{ segundos} \rightarrow 34{,}048 \text{ años} \qquad \text{(Expresión 6.4)}$$

Estos ejemplos ilustran perfectamente la problemática de tener que procesar por fuerza bruta un conjunto de atributos de un tamaño relativamente 'grande' (existen problemas reales donde el número de atributos podrían llegar a ser mucho más de 30). Se hace necesario investigar otras técnicas que permitan realizar este tipo de cálculos en un tiempo razonable.

Una posibilidad sería la utilización de técnicas computación evolutiva o algoritmos genéticos, que ya se apuntaron, en combinación de la métrica 'ACI', en el trabajo de (Zaragozí et al. 2011). Estas técnicas están inspiradas en el mecanismo de evolución biológica observado en los seres vivos. A grosso modo, consiste en considerar una combinación de atributos como un '*individuo*' que dispone de un '*cromosoma*' formado por una combinación de dígitos que pueden tomar los valores cero o uno; cada uno de

estos dígitos sería un '*gen*' del cromosoma. Cada gen estará asociado a uno de los atributos del conjunto de datos a procesar. Un valor de '0' significa que el atributo correspondiente no se debe considerar; por el contrario, un valor de '1' en el gen se interpretará como que dicho atributo sí va a ser considerado. Inicialmente se dispondrá de una '*población*' de individuos que deben '*evolucionar*'. La '*evolución*' consiste en ir combinando los cromosomas de los individuos de dicha población y, adicionalmente, hacerlos '*mutar*', para generar nuevos individuos. Se verificará en cada caso el valor de la métrica 'WACI' para cada uno de los nuevos individuos. Finalmente se promocionará a los mejores (los que tengan mejor valor de 'WACI') y penalizará a los peores.

Este tipo de técnicas tiene el inconveniente de que no garantiza la obtención de una solución óptima (al contrario que la fuerza bruta), ya que no prueba todas las combinaciones. Sin embargo, sí que proporcionan una solución subóptima, y, lo más importante, pueden dar dicha solución en un tiempo razonable.

Otra opción para mejorar los tiempos de proceso en el problema de extracción de características son las técnicas de supercomputación, más concretamente, la computación paralela. Esta metodología consiste en dividir un problema en partes independientes y ejecutar cada una de ellas simultáneamente en procesadores diferentes. Idealmente (aunque en la práctica se sabe que no es exactamente así), si se dispone de un número 'p' de procesadores para ejecutar un proceso paralelo, el tiempo de ejecución se dividiría por ese valor 'p'. El problema de la extracción de características con el método CREA y la métrica 'WACI' (o también con 'ACI') consiste en hacer múltiples ejecuciones del algoritmo CREA, cada una de ellas es independiente del resto por lo que, en principio, se trata de un proceso susceptible de ser paralelizado.

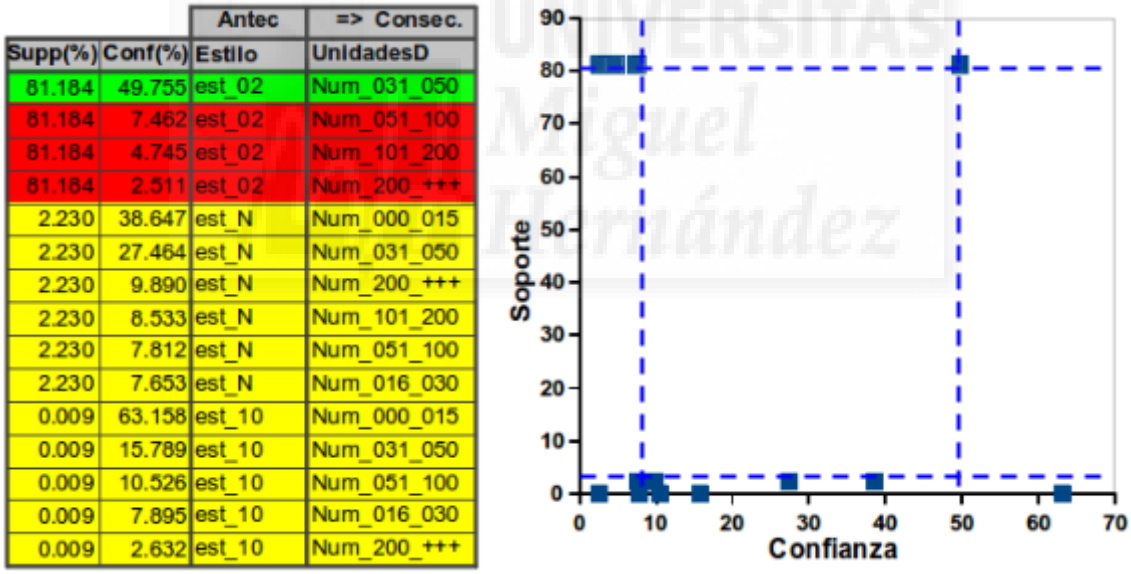
Los algoritmos genéticos y la computación paralela no son métodos excluyentes. En problemas de especial envergadura podría plantearse una solución que combine ambas técnicas por lo que la investigación de ambos campos, tanto por separado como conjuntamente, aportará importantes mejoras a la problemática de la extracción de características sobre conjuntos de datos muy grandes, especialmente, cuando dichos conjuntos tienen un elevado número de columnas.

6.4.2. Mejorar la presentación de los sistemas de reglas

Pese a la reducción de reglas que realiza el método RBS (incorporado en el algoritmo CREA), en ocasiones, el número de reglas que proporciona este método puede ser algo elevado para que un analista humano pueda gestionarlo. Como se comentó en el epígrafe 2.1.2, un conjunto de reglas muy numeroso puede servir para construir un sistema experto, ya que este puede procesar automáticamente las reglas y generar conclusiones. Este trabajo se ha realizado pensando más bien en la implementación de sistemas de apoyo a la toma de decisiones, en los que la información procesada se presenta a un experto humano que, en base a dicha información y a su experiencia,

ejercerá la responsabilidad última de tomar las decisiones que considere más beneficiosas para su organización. Si el modelo resultante, pese a la reducción del método RBS, contiene muchas reglas, el experto humano tendrá dificultades para manejarlo, por lo que es de vital importancia que estos modelos vayan acompañados de algún mecanismo que permita al usuario '*navegar*' interactivamente por las reglas del modelo. Actualmente se dispone de dos sistemas de representación de los modelos de reglas de clasificación que ayudan al usuario-analista a visualizar el modelo.

El primero de ellos consiste en una representación tabular en la que las reglas se representan como en las tablas 5.9, 5.10 y siguientes del epígrafe 5.3.2. La información de cada regla se representa en una fila de la tabla, los dos primeros campos son los valores de soporte y confianza, y el resto son los valores que toman los distintos atributos para esa regla. El valor final (a la derecha) es el consecuente de la regla considerada. En esta representación tabular, las reglas están ordenadas por el valor 'rs' de cada una de ellas, de tal forma que la primera fila contiene la regla más confiable (mayor valor de 'rs'). Además, para identificar la región a la que pertenece cada regla, se colorean en verde las de 'REG-2' (reglas confiables), en rojo las de 'REG-1' (reglas de descarte) y en amarillo las de 'REG-3' (reglas a observar).



		Antec	=> Consec.
Supp(%)	Conf(%)	Estilo	UnidadesD
81.184	49.755	est_02	Num_031_050
81.184	7.462	est_02	Num_051_100
81.184	4.745	est_02	Num_101_200
81.184	2.511	est_02	Num_200_+++
2.230	38.647	est_N	Num_000_015
2.230	27.464	est_N	Num_031_050
2.230	9.890	est_N	Num_200_+++
2.230	8.533	est_N	Num_101_200
2.230	7.812	est_N	Num_051_100
2.230	7.653	est_N	Num_016_030
0.009	63.158	est_10	Num_000_015
0.009	15.789	est_10	Num_031_050
0.009	10.526	est_10	Num_051_100
0.009	7.895	est_10	Num_016_030
0.009	2.632	est_10	Num_200_+++

Figura 6.1: Vista tabular y gráfica de un sistema de reglas

La segunda forma de visualizar un sistema de reglas consiste en una representación gráfica real, al estilo de las que se muestran en el epígrafe 2.1.2 (ver figura 2.6 y 2.7), pero eliminando las reglas de la región 'REG-0'. En dichas gráficas se representa cada regla como un punto cuyas coordenadas son sus valores de soporte y confianza, de tal forma que la regla que queda más a la derecha y más arriba será la más confiable de todas.

Se ha desarrollado un entorno interactivo en el cual el analista puede aplicar ciertos filtros (ver figura 6.2) sobre un conjunto de reglas para, en un momento dado, poder

hacer preguntas al modelo del estilo '*¿Cómo se comporta la variable consecuente cuando el antecedente toma un determinado valor?*'.

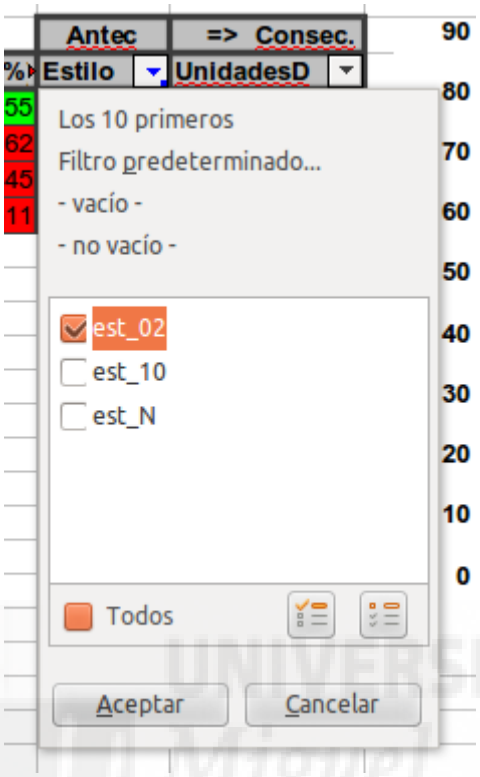
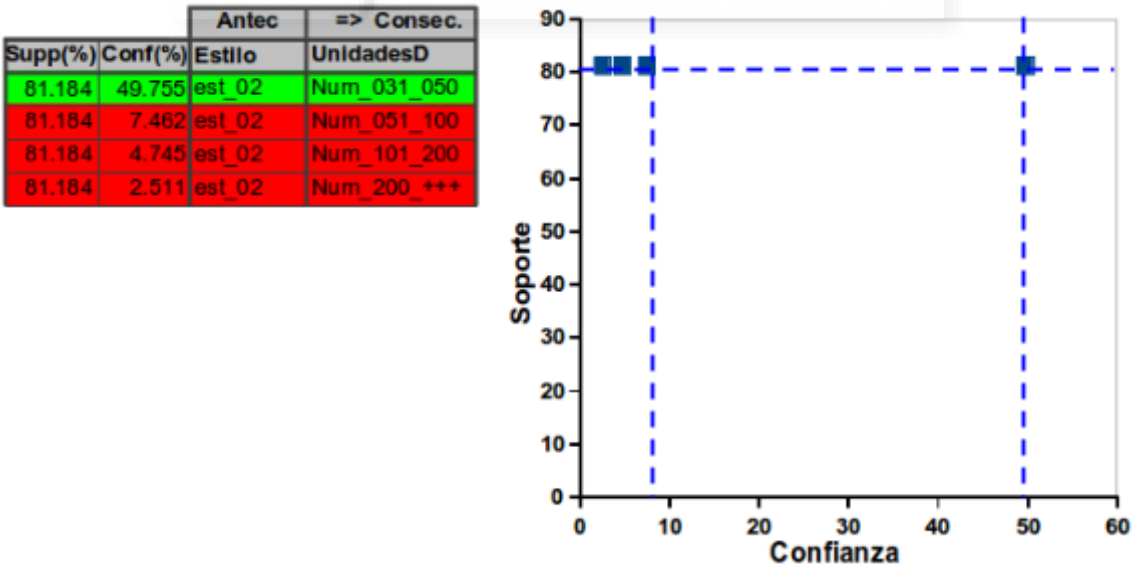


Figura 6.2: Filtro para un sistema de reglas (implementado en una hoja de cálculo)



		Antec	=> Consec.
Supp(%)	Conf(%)	Estilo	UnidadesD
81.184	49.755	est_02	Num_031_050
81.184	7.462	est_02	Num_051_100
81.184	4.745	est_02	Num_101_200
81.184	2.511	est_02	Num_200_+++

Figura 6.3: Vista de un sistema de reglas tras aplicar un filtro

Aplicado el filtro de la figura 6.2 al modelo representado en la figura 6.1, las vistas tabular y gráfica del modelo se modifican como se muestra en la figura 6.3.

Esta forma de representar los sistemas de reglas proporciona al analista un método de una cierta potencia para poder estudiar el modelo generado, aunque sigue pareciendo insuficiente. En el ámbito del análisis de grandes volúmenes de datos y, concretamente, en la Inteligencia de Negocios, existe un campo de investigación abierto que es la visualización de los resultados tras cualquier proceso de análisis. Como ya se ha comentado anteriormente los ejecutivos valoran el hecho de poder tomar decisiones lo más rápidamente posible. Un requisito para ello es que los procesos de análisis sean lo más eficientes posibles en cuanto al tiempo de ejecución; pero otro requisito igualmente importante es que la información devuelta sea fácilmente interpretable. De poco serviría un proceso de cálculo muy rápido si la información resultante que proporciona necesita que el analista le tenga que dedicar horas de estudio. Una posible forma de representación gráfica de fácil interpretación sería dibujar el árbol de decisión que se genera al aplicar el método ID3 (ver figuras 2.1 y 2.2). Este tipo de gráficos tiene dos importantes inconvenientes: Por una parte el dibujo de un árbol de decisión puede ser muy extenso cuando el número de reglas es elevado; en ese caso el árbol sería difícil de visualizar en una pantalla de ordenador, mucho más en un dispositivo móvil con tamaños de pantalla reducidos. El otro inconveniente que tienen los árboles de clasificación es que el nodo raíz, único punto de entrada al mismo, se refiere a una única variable predeterminada por el método de generación del árbol (ID3 u otro), en este caso un analista está obligado a analizar el modelo empezando siempre por la misma variable y, en general, tendrá poca flexibilidad para recorrer el árbol ya que a partir de un nodo tendrá un número de caminos reducido por los que seguir su análisis.

Una alternativa a los árboles de decisión que se puede investigar son los grafos de decisión (o '*decision graphs*'), de los cuales, los árboles de decisión se pueden considerar un subconjunto. La idea es generar vistas del modelo de reglas que el analista pueda recorrer de la forma más flexible posible. Nótese que la vista tabular de un sistema de reglas (figuras 6.1 y 6.3) permite esta flexibilidad, a la hora de recorrer las reglas. Una solución interesante consistiría en dar una representación gráfica en forma de grafo al conjunto de reglas, actualizándolo cada vez que se modifica el filtro aplicado.

En general las posibilidades de representar gráficamente un conjunto de reglas pueden ser muy variadas. La idea que debe primar en cualquier estudio y desarrollo sobre este particular es que la representación escogida resulte útil y clara para el analista que tiene que tomar decisiones en base a la información representada.

6.4.3. Actualización de modelos con nuevos datos

Actualmente, tal y como está diseñado el método CREA, con él es posible generar un modelo a partir de un conjunto de datos de entrada. Con el paso del tiempo dicho conjunto de datos crece (sea cual sea la ventana temporal considerada). Si se desea actualizar el modelo anterior con los nuevos datos, la única opción (con las técnicas

desarrolladas en esta tesis) es añadirlos al conjunto de datos original y volver a generar un modelo más actualizado, desechando el que quedó obsoleto.

En el campo de la Minería de Datos, existe un rama denominada '*Data Stream Mining*' (o 'Minado de Flujos de Datos') que investiga la forma de disponer de modelos de representación del conocimiento actualizados a medida que se tienen nuevos ejemplos con los que seguir alimentando el modelo. Con estas técnicas, el modelo original nunca se desecha, lo que se hace es actualizarlo con los nuevos registros de información que se capturan. Esta tarea de actualización del modelo puede realizarse con la frecuencia que se considere más conveniente. En el caso más extremo, cada nuevo registro de datos se utilizaría en el mismo momento de ser capturado para renovar el sistema. En este caso se tendría un modelo actualizado en tiempo real. Otra opción sería establecer una determinada ventana temporal o de número de nuevos registros disponibles, y ejecutar un proceso por lotes para poner al día el modelo.

Este tipo de técnicas se podría aplicar a los sistemas de reglas generados con el método CREA actualizando los valores de soporte y confianza de las reglas disponibles o introduciendo nuevas reglas cuando no estuvieran presentes en un modelo anterior.

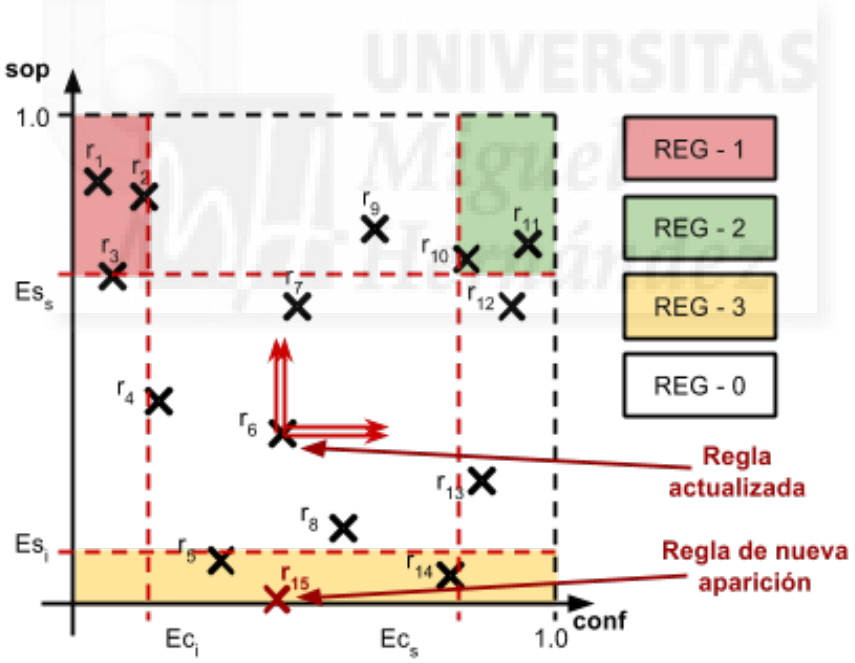


Figura 6.4: Actualización del modelo de reglas

En la figura 6.4 se ilustra cómo podría realizarse la actualización de un modelo de reglas generado con el método CREA. Si se dispone de un nuevo ejemplo, compatible con la regla 'r_6' se deberá actualizar el valor de soporte y confianza de esta regla. Si el flujo de datos trae un ejemplo que no es compatible con ninguna de las reglas del modelo, entonces se introduce una nueva regla ('r_{15}') que, por ser la primera vez que se da en el conjunto de datos, tendrá un soporte mínimo. Una vez actualizadas las reglas únicamente quedará recalcular los ejes y ver si alguna de las reglas ha cambiado de región.

6.4.4. Aplicación del algoritmo CREA a otros ámbitos

Desde el punto de vista de la aplicación de los sistemas de reglas de clasificación al ámbito de la Inteligencia de Negocio, el siguiente paso natural para avanzar en esta investigación, pasa necesariamente por buscar otros marcos de aplicación empresarial, donde testear el algoritmo CREA y sus futuras actualizaciones. Así, por ejemplo, ya se ha iniciado el análisis predictivo de portales de venta on-line, o el modelado del comportamiento de clientes en portales de reserva telemática, entre otros, con resultados muy satisfactorios.

Anexo I
Análisis multidimensional del tiempo de procesamiento

AI.1. Cubo de información

La variable objeto de estudio empírico, el tiempo de ejecución del algoritmo CREA, depende del conjunto de datos de ejemplos ,'E', que en cada caso se quiere procesar, el cual viene definido por tres parámetros que lo caracterizan, el número de ejemplos o filas a procesar, 'N'; el número de atributos de cada ejemplo, es decir, las columnas del conjunto de datos, 'C'; y el número de valores, 'V', que puede tomar cada atributo. En el presente estudio, para poder sistematizar el análisis, el valor 'V' se considera constante para todos los atributos de un mismo conjunto de datos.

Se han simulado 450 conjuntos de datos donde los parámetros 'N', 'C' y 'V' toman los siguientes valores:

$$N \in \{2x10^5, 4x10^5, 6x10^5, 8x10^5, 10x10^5, 12x10^5, 14x10^5, 16x10^5, 18x10^5, 20x10^5\}$$

$$C \in \{2, 4, 8, 16, 32\}$$

$$V \in \{2, 3, 4, 6, 8, 10, 12, 14, 16\}$$

Es decir, el parámetro ‘N’ puede tomar 10 valores distintos, ‘C’ puede tomar 5 valores y ‘V’ puede variar entre 9 posibles valores. De todas las combinaciones posibles de los eventuales valores de estas variables surgen los 450 conjuntos de datos que se han considerado para realizar el estudio (|N| x |C| x |V| = 450). Con cada uno de estos conjuntos de datos se ha ejecutado el algoritmo CREA, realizando la medición del tiempo de procesamiento. Dado que los datos obtenidos en estas mediciones están en función de los parámetros ‘N’, ‘C’ y ‘V’, se puede plantear la construcción de un cubo de datos de 3 dimensiones (ver figura AI.1) para realizar un análisis multidimensional de los tiempos de procesamiento del algoritmo CREA.

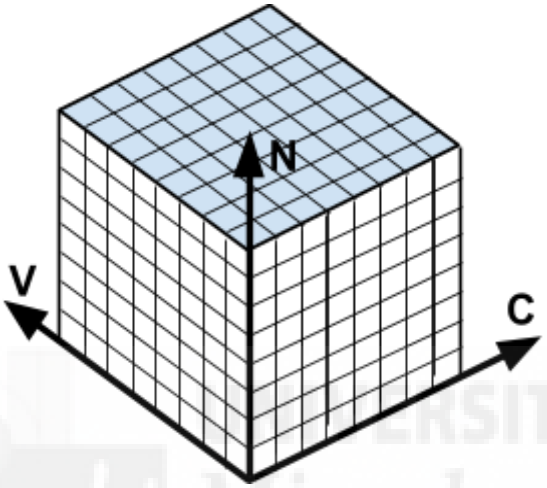


Figura AI.1: Representación gráfica de un cubo de datos de tres dimensiones

Físicamente, el citado cubo de datos se puede implementar como un cubo OLAP con diseño en estrella, donde la tabla central sería la tabla de hechos del cubo, esta contendrá los datos procedentes de las mediciones que se quiere analizar, es decir, los tiempos de procesamiento. Las dimensiones de dicho cubo estarán representadas por tres tablas para los posibles valores de cada uno de los parámetros de los que dependen los tiempos de procesamiento, ‘N’, ‘C’ y ‘V’.

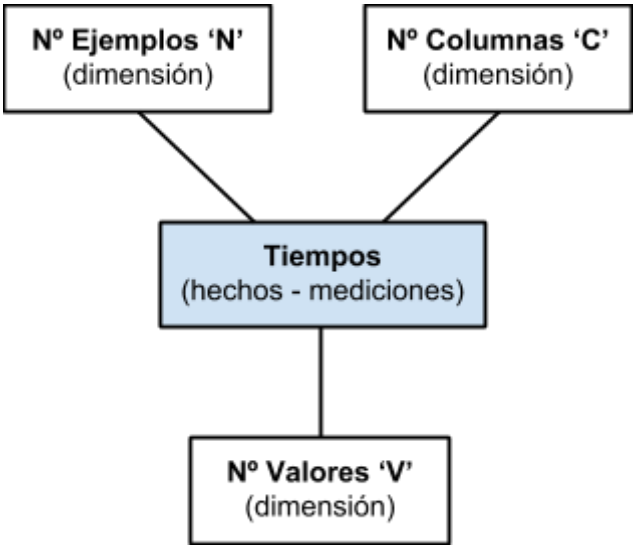


Figura AI.2: Diseño en estrella del cubo de datos de tiempos

Con este planteamiento resulta sencillo proponer diferentes formas de presentar la evolución de la variable 'tiempo' en función de los parámetros 'N', 'C' y 'V' de los que depende y así poder hacer un análisis más exhaustivo.

Tabla AI.1: Vista { C , V } frente a { N } de los tiempos de ejecución

C	V	N= $2x10^5$	N= $4x10^5$	N= $6x10^5$	N= $8x10^5$	N= $10x10^5$	N= $12x10^5$	N= $14x10^5$	N= $16x10^5$	N= $18x10^5$	N= $20x10^5$
2	2	21.9	42.2	62.5	82.8	104.7	123.4	145.3	165.7	185.9	206.2
2	3	26.5	51.5	78.1	103.1	129.6	154.6	179.7	207.8	232.8	257.8
2	4	31.2	60.9	92.2	123.5	154.7	184.4	217.2	246.9	278.1	307.8
2	6	40.6	81.2	121.8	162.5	203.2	245.3	286.0	328.1	368.7	410.9
2	8	50.0	101.6	153.1	201.5	253.1	303.1	354.7	406.2	457.8	507.8
2	10	60.9	121.9	181.2	242.2	301.6	362.5	423.4	486.0	545.3	607.8
2	12	71.9	143.7	214.0	286.0	357.8	426.6	500.0	571.9	642.2	717.2
2	14	82.8	165.6	246.9	329.7	411.0	493.8	576.6	657.8	740.7	826.5
2	16	93.7	185.9	279.7	371.9	464.1	556.3	648.4	742.2	835.9	929.6
4	2	85.9	173.4	259.4	348.4	435.9	521.9	609.4	693.7	782.8	870.3
4	3	106.3	215.6	325.0	434.4	546.9	654.6	768.7	876.5	987.5	1100.0
4	4	123.4	251.5	384.3	510.9	671.9	778.1	940.7	1082.8	1203.1	1334.3
4	6	160.9	314.1	470.3	626.5	865.6	951.6	1193.8	1446.9	1523.5	1864.1
4	8	206.3	390.7	581.3	768.8	967.2	1145.3	1345.3	1584.4	1739.1	2017.2
4	10	286.0	518.8	751.6	984.3	1228.2	1473.4	1720.3	1984.4	2209.4	2504.7
4	12	409.4	675.0	957.8	1214.1	1485.9	1743.8	2004.7	2284.3	2551.5	2895.3
4	14	579.6	892.1	1215.6	1529.7	1836.0	2121.8	2400.0	2678.1	2984.4	3853.1
4	16	873.5	1292.2	1623.4	1971.9	2348.4	2704.7	3029.7	3385.9	3740.6	4081.3
8	2	356.2	742.2	1143.8	1454.7	1803.1	2175.0	2559.4	2921.8	3400.0	3693.7
8	3	493.7	1079.7	1589.1	2121.9	2715.6	3190.6	3779.7	4337.5	4887.5	5475.0
8	4	1139.1	1871.9	2548.4	3243.8	3995.3	4635.9	5386.0	6084.4	6714.0	7543.8
8	6	2592.2	4846.9	6782.8	9262.5	12661.0	15840.7	18390.7	15682.8	16901.6	18626.6
8	8	3039.1	5911.0	8765.6	11881.3	14517.2	16981.2	19621.9	22870.3	25232.9	27082.8
8	10	3623.4	7178.1	9773.4	14190.6	16042.2	20476.5	24728.1	28907.8	31031.3	34503.1
8	12	3543.8	7451.6	10486.0	15240.7	16406.2	21496.9	26257.8	30632.8	28815.7	33482.8
8	14	3592.2	7512.5	10937.5	15931.3	17378.1	22814.1	27834.4	33500.0	30514.1	35579.6
8	16	3746.9	7729.7	11017.2	16425.0	18003.1	23561.0	29109.4	35428.2	32306.3	37560.9
16	2	2961.0	5690.6	7904.7	10278.1	12220.3	14364.1	16687.5	19207.8	21746.8	24231.3
16	3	4564.0	10354.6	14387.5	24481.2	29262.5	33265.6	45817.2	59315.6	75893.7	64739.1
16	4	4231.3	9300.0	17284.4	20937.5	29150.0	39946.9	52575.0	46112.5	51581.3	62971.9
16	6	4390.6	9157.8	14093.8	19545.3	26634.4	35192.2	36668.8	43014.1	46409.3	55281.2
16	8	4382.8	9521.8	15398.5	20534.3	25409.4	30795.4	34679.7	40021.9	45426.6	51703.1
16	10	5456.2	10726.6	16198.4	22057.8	28173.5	35743.7	42332.9	49861.0	56236.0	62960.9
16	12	5435.9	12235.9	18570.3	24479.7	30231.3	36056.3	40965.7	47335.9	54375.0	60689.1
16	14	5989.1	11914.1	19436.0	27309.4	34656.3	42089.1	48403.1	55364.1	62090.7	68067.2
16	16	7239.1	12782.8	19184.3	26917.2	34836.0	43226.5	51579.7	60587.5	69168.8	78279.7
32	2	10381.2	24142.2	34725.0	54681.2	59109.4	80293.8	103779.7	130282.8	129020.3	136096.9
32	3	9246.9	21984.4	31412.5	51523.4	54103.1	72500.0	95192.2	120854.7	103211.0	120107.8
32	4	9296.9	20646.9	28576.5	47801.6	69915.7	63828.1	83643.7	106926.6	130068.7	156540.6
32	6	9432.8	19364.1	29368.7	45698.5	51856.2	68189.1	87121.8	105979.7	100217.2	112842.2
32	8	8443.7	18787.5	30290.6	40454.7	50653.1	60165.6	70648.4	80859.4	89837.5	100078.1
32	10	10832.8	21506.3	31565.6	43417.2	56071.9	70656.3	85953.2	101003.1	114484.4	128120.3
32	12	10207.8	24414.0	38435.9	49787.5	61879.7	72765.6	84560.9	92729.7	105367.2	118275.0
32	14	11490.6	22843.8	38571.9	55395.3	72539.1	88464.0	103993.8	116043.7	129711.0	142379.7
32	16	14615.6	25098.4	37075.0	52078.2	69934.3	88725.0	108309.4	126728.1	146712.5	166485.9

Tabla AI.2: Vista { V , C } frente a { N } de los tiempos de ejecución

V	C	N= $2x10^5$	N= $4x10^5$	N= $6x10^5$	N= $8x10^5$	N= $10x10^5$	N= $12x10^5$	N= $14x10^5$	N= $16x10^5$	N= $18x10^5$	N= $20x10^5$
2	2	21.9	42.2	62.5	82.8	104.7	123.4	145.3	165.7	185.9	206.2
2	4	85.9	173.4	259.4	348.4	435.9	521.9	609.4	693.7	782.8	870.3
2	8	356.2	742.2	1143.8	1454.7	1803.1	2175.0	2559.4	2921.8	3400.0	3693.7
2	16	2961.0	5690.6	7904.7	10278.1	12220.3	14364.1	16687.5	19207.8	21746.8	24231.3
2	32	10381.2	24142.2	34725.0	54681.2	59109.4	80293.8	103779.7	130282.8	129020.3	136096.9
3	2	26.5	51.5	78.1	103.1	129.6	154.6	179.7	207.8	232.8	257.8
3	4	106.3	215.6	325.0	434.4	546.9	654.6	768.7	876.5	987.5	1100.0
3	8	493.7	1079.7	1589.1	2121.9	2715.6	3190.6	3779.7	4337.5	4887.5	5475.0
3	16	4564.0	10354.6	14387.5	24481.2	29262.5	33265.6	45817.2	59315.6	75893.7	64739.1
3	32	9246.9	21984.4	31412.5	51523.4	54103.1	72500.0	95192.2	120854.7	103211.0	120107.8
4	2	31.2	60.9	92.2	123.5	154.7	184.4	217.2	246.9	278.1	307.8
4	4	123.4	251.5	384.3	510.9	671.9	778.1	940.7	1082.8	1203.1	1334.3
4	8	1139.1	1871.9	2548.4	3243.8	3995.3	4635.9	5386.0	6084.4	6714.0	7543.8
4	16	4231.3	9300.0	17284.4	20937.5	29150.0	39946.9	52575.0	46112.5	51581.3	62971.9
4	32	9296.9	20646.9	28576.5	47801.6	69915.7	63828.1	83643.7	106926.6	130068.7	156540.6
6	2	40.6	81.2	121.8	162.5	203.2	245.3	286.0	328.1	368.7	410.9
6	4	160.9	314.1	470.3	626.5	865.6	951.6	1193.8	1446.9	1523.5	1864.1
6	8	2592.2	4846.9	6782.8	9262.5	12661.0	15840.7	18390.7	15682.8	16901.6	18626.6
6	16	4390.6	9157.8	14093.8	19545.3	26634.4	35192.2	36668.8	43014.1	46409.3	55281.2
6	32	9432.8	19364.1	29368.7	45698.5	51856.2	68189.1	87121.8	105979.7	100217.2	112842.2
8	2	50.0	101.6	153.1	201.5	253.1	303.1	354.7	406.2	457.8	507.8
8	4	206.3	390.7	581.3	768.8	967.2	1145.3	1345.3	1584.4	1739.1	2017.2
8	8	3039.1	5911.0	8765.6	11881.3	14517.2	16981.2	19621.9	22870.3	25232.9	27082.8
8	16	4382.8	9521.8	15398.5	20534.3	25409.4	30795.4	34679.7	40021.9	45426.6	51703.1
8	32	8443.7	18787.5	30290.6	40454.7	50653.1	60165.6	70648.4	80859.4	89837.5	100078.1
10	2	60.9	121.9	181.2	242.2	301.6	362.5	423.4	486.0	545.3	607.8
10	4	286.0	518.8	751.6	984.3	1228.2	1473.4	1720.3	1984.4	2209.4	2504.7
10	8	3623.4	7178.1	9773.4	14190.6	16042.2	20476.5	24728.1	28907.8	31031.3	34503.1
10	16	5456.2	10726.6	16198.4	22057.8	28173.5	35743.7	42332.9	49861.0	56236.0	62960.9
10	32	10832.8	21506.3	31565.6	43417.2	56071.9	70656.3	85953.2	101003.1	114484.4	128120.3
12	2	71.9	143.7	214.0	286.0	357.8	426.6	500.0	571.9	642.2	717.2
12	4	409.4	675.0	957.8	1214.1	1485.9	1743.8	2004.7	2284.3	2551.5	2895.3
12	8	3543.8	7451.6	10486.0	15240.7	16406.2	21496.9	26257.8	30632.8	28815.7	33482.8
12	16	5435.9	12235.9	18570.3	24479.7	30231.3	36056.3	40965.7	47335.9	54375.0	60689.1
12	32	10207.8	24414.0	38435.9	49787.5	61879.7	72765.6	84560.9	92729.7	105367.2	118275.0
14	2	82.8	165.6	246.9	329.7	411.0	493.8	576.6	657.8	740.7	826.5
14	4	579.6	892.1	1215.6	1529.7	1836.0	2121.8	2400.0	2678.1	2984.4	3853.1
14	8	3592.2	7512.5	10937.5	15931.3	17378.1	22814.1	27834.4	33500.0	30514.1	35579.6
14	16	5989.1	11914.1	19436.0	27309.4	34656.3	42089.1	48403.1	55364.1	62090.7	68067.2
14	32	11490.6	22843.8	38571.9	55395.3	72539.1	88464.0	103993.8	116043.7	129711.0	142379.7
16	2	93.7	185.9	279.7	371.9	464.1	556.3	648.4	742.2	835.9	929.6
16	4	873.5	1292.2	1623.4	1971.9	2348.4	2704.7	3029.7	3385.9	3740.6	4081.3
16	8	3746.9	7729.7	11017.2	16425.0	18003.1	23561.0	29109.4	35428.2	32306.3	37560.9
16	16	7239.1	12782.8	19184.3	26917.2	34836.0	43226.5	51579.7	60587.5	69168.8	78279.7
16	32	14615.6	25098.4	37075.0	52078.2	69934.3	88725.0	108309.4	126728.1	146712.5	166485.9

Las tablas AI.1 y AI.2 muestran de dos formas diferentes los mismos datos de tiempos resultantes de las mediciones realizadas. Son, por tanto, dos vistas posibles del mismo cubo de información.

En los siguientes apartados se va a presentar un análisis exhaustivo de la evolución de la variable ‘tiempo’ en función de las dimensiones ‘N’, ‘C’ y ‘V’; se plantean tres tipos de análisis en los que, de las tres dimensiones, dos se fijan a un determinado valor y la tercera se utiliza para observar la evolución de los tiempos de procesamiento. En cada caso se presentan gráficas que representan un subconjunto de datos extraído de alguna de las tablas AI.1 o AI.2 a las que en adelante se denominará ‘vista 1’ y ‘vista 2’ del cubo de datos, y que de forma esquemática se van a representar como en la figura AI.3. Nótese que la ‘vista 1’ se divide en 5 bloques (uno por cada valor de ‘C’) y en cada

bloque hay 9 filas de datos (una por cada valor de ‘V’). En la ‘vista 2’ los datos se agrupan en 9 bloques de 5 filas cada uno (valores de ‘V’ y ‘C’ respectivamente). En ambos casos cada bloque de datos está estructurado en 10 columnas (uno por cada valor de ‘N’).

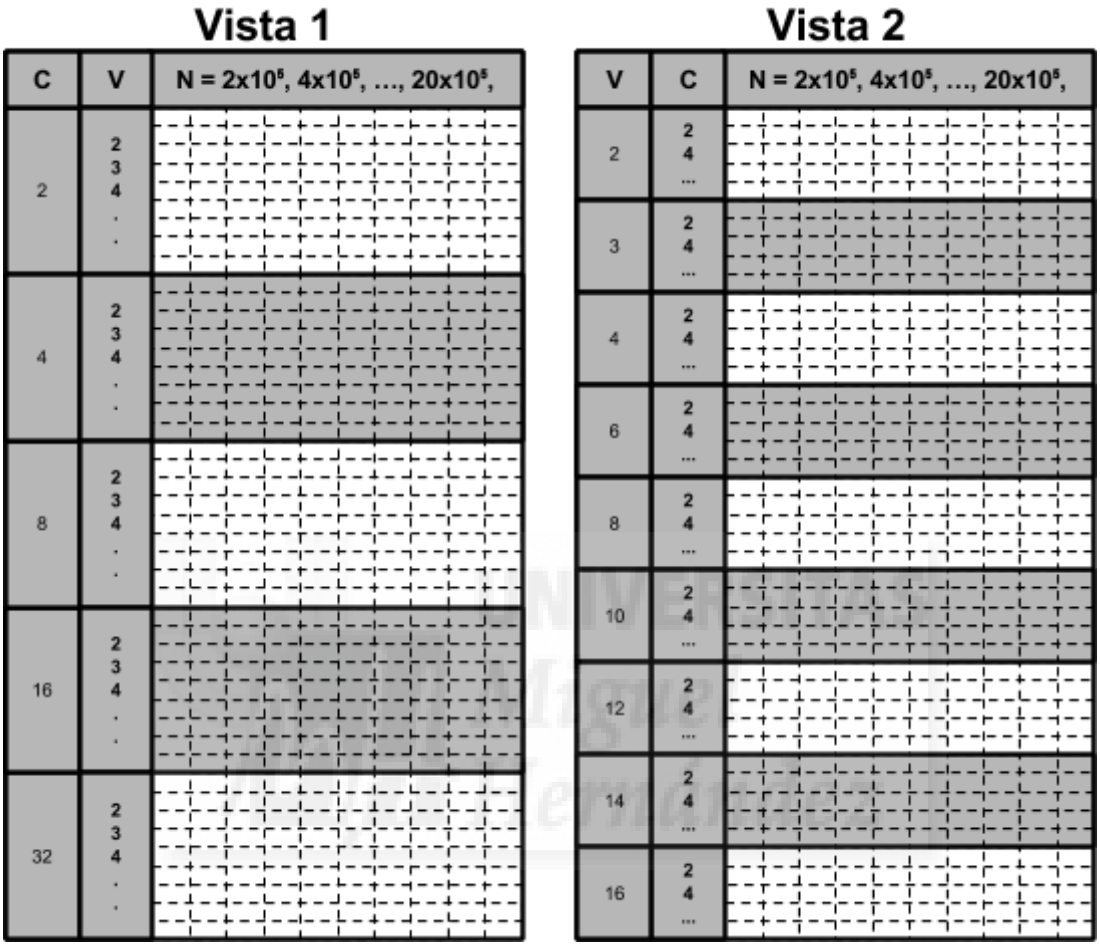


Figura AI.3: Representación esquemática de las vistas del cubo de datos ‘Vista 1’ y ‘Vista 2’

AI.2. Análisis en función del número de ejemplos ‘N’

AI.2.1. Comparativa por número de columnas ‘C’

Este apartado contiene 5 gráficas extraídas de los datos de la ‘vista 1’, una por cada valor de ‘C’. En cada gráfica se representan 9 series de datos que se corresponden con las filas de cada uno de los 5 bloques de dicha vista. El eje X representa el número de ejemplos ‘N’ del conjunto de datos procesado.

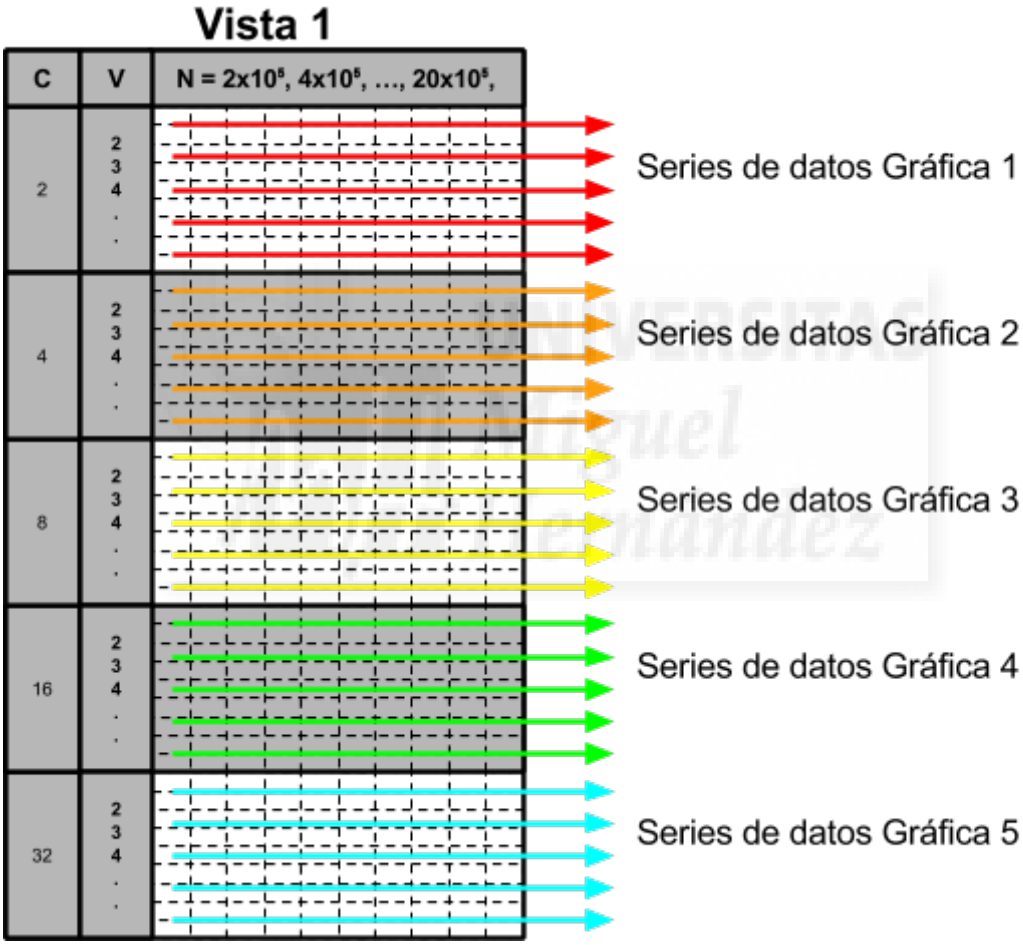
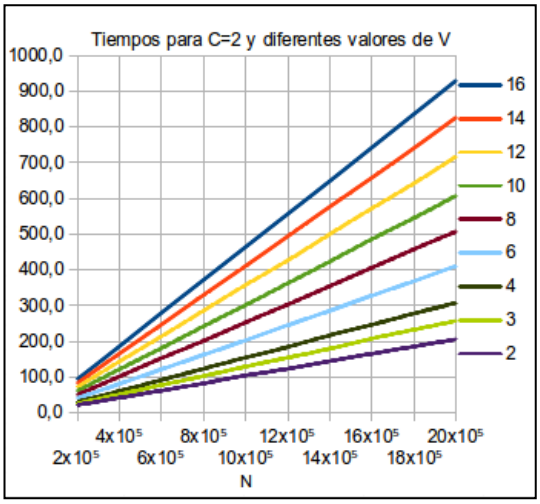
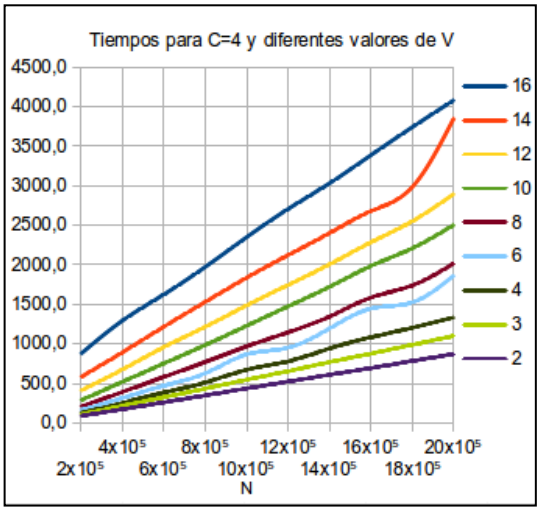


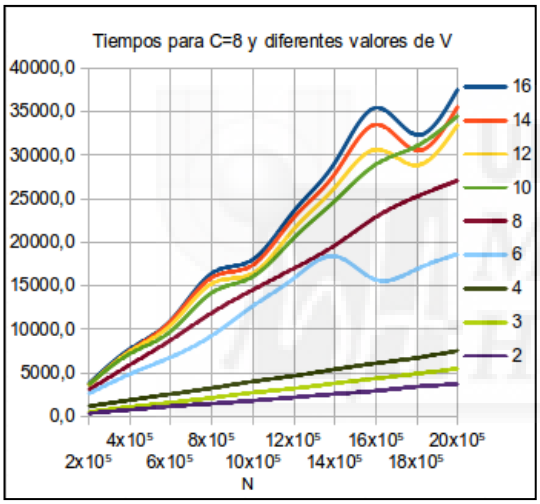
Figura AI.4: Origen de las series de datos de las gráficas del epígrafe AI.1.1



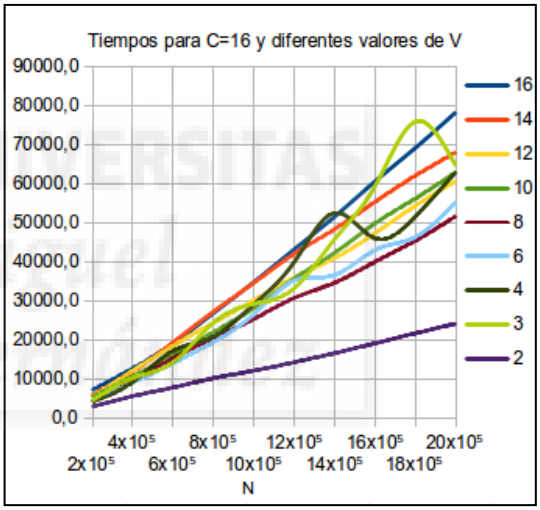
GRÁFICA 1: Tiempo frente a 'N' para 'C=2'



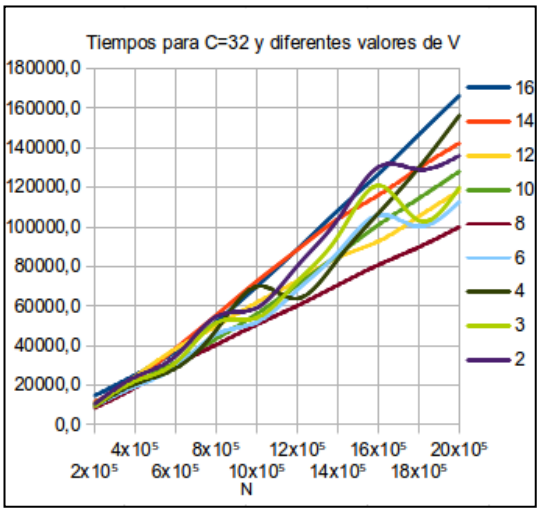
GRÁFICA 2: Tiempo frente a 'N' para 'C=4'



GRÁFICA 3: Tiempo frente a 'N' para 'C=8'



GRÁFICA 4: Tiempo frente a 'N' para 'C=16'



GRÁFICA 5: Tiempo frente a 'N' para 'C=32'

AI.2.2. Comparativa por número de valores ‘V’

Este apartado contiene 9 gráficas extraídas de los datos de la ‘vista 2’, una por cada valor de ‘V’. En cada gráfica se representan 5 series de datos que se corresponden con las filas de cada uno de los 9 bloques de dicha vista. El eje X representa el número de ejemplos ‘N’ del conjunto de datos procesado.

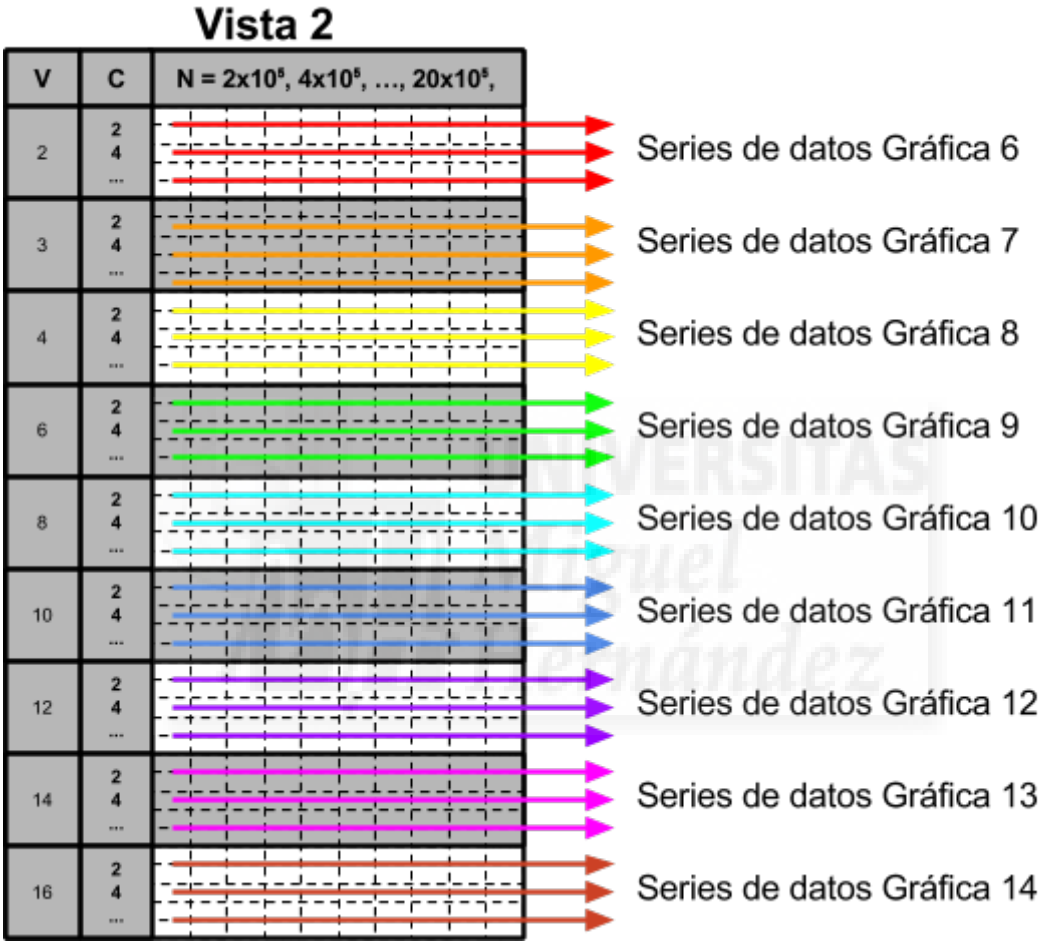
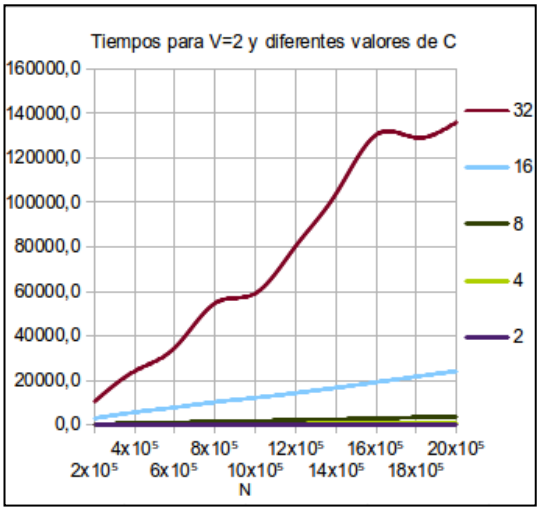
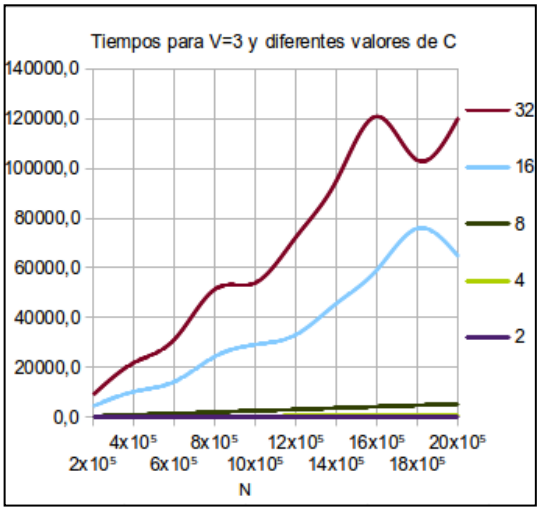


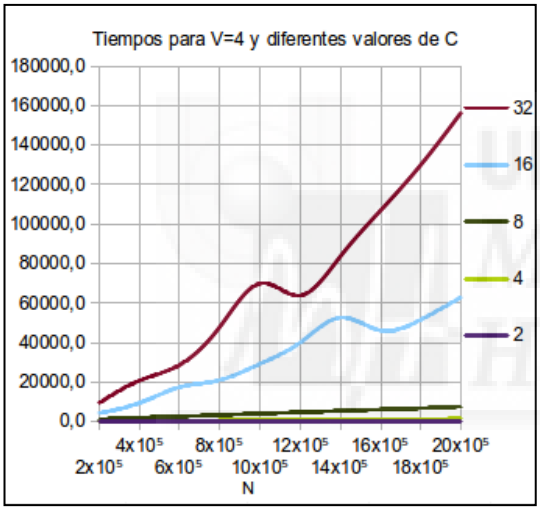
Figura AI.5: Origen de las series de datos de las gráficas del epígrafe AI.1.2



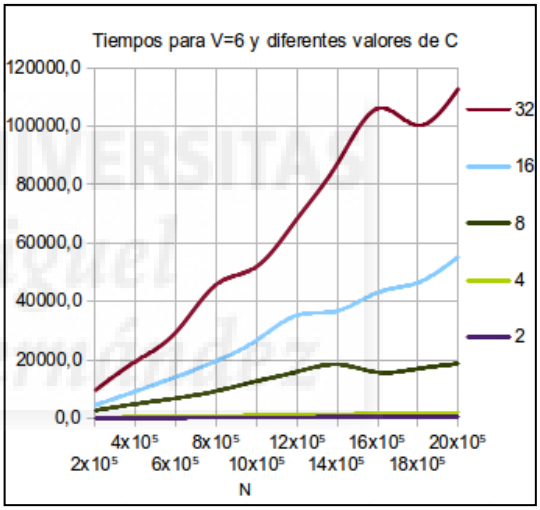
GRÁFICA 6: Tiempo frente a 'N' para 'V=2'



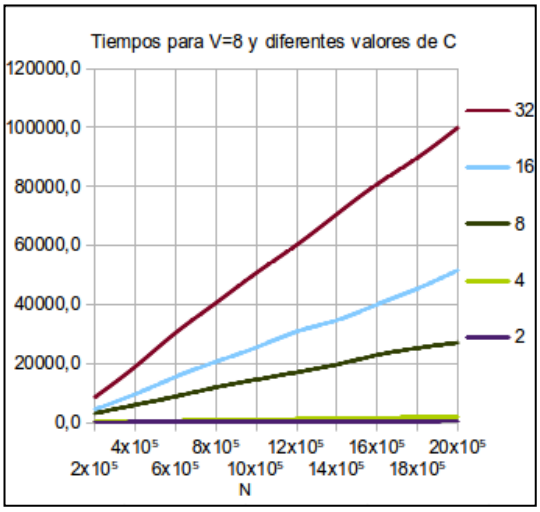
GRÁFICA 7: Tiempo frente a 'N' para 'V=3'



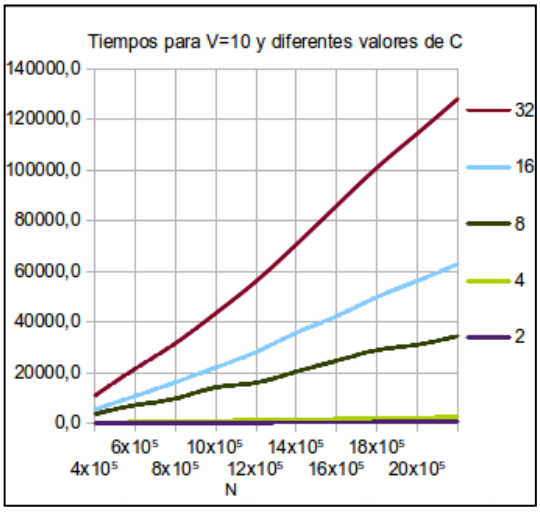
GRÁFICA 8: Tiempo frente a 'N' para 'V=4'



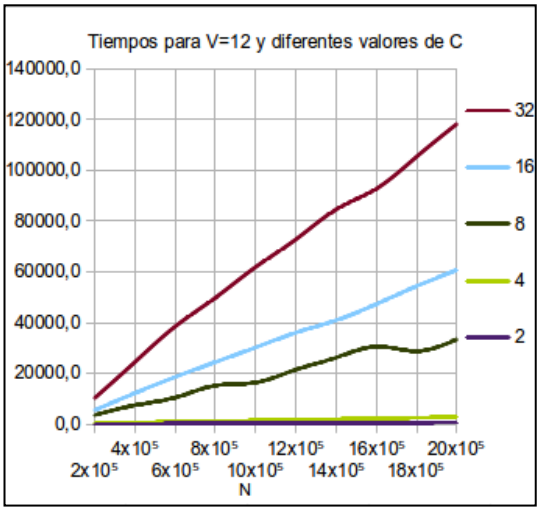
GRÁFICA 9: Tiempo frente a 'N' para 'V=6'



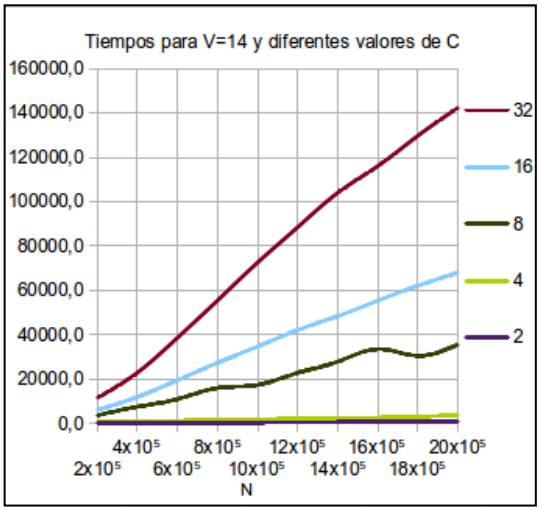
GRÁFICA 10: Tiempo frente a 'N' para 'V=8'



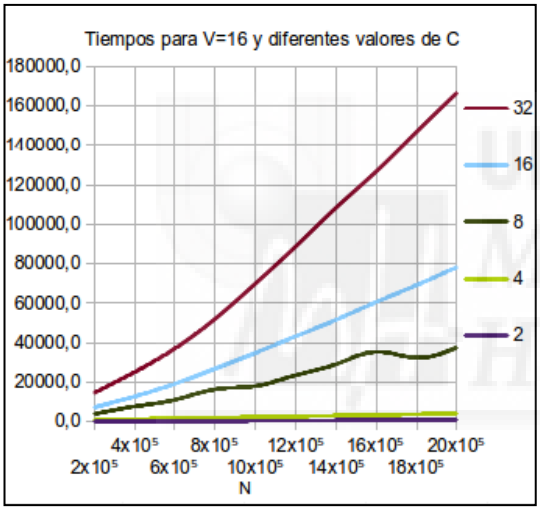
GRÁFICA 11: Tiempo frente a 'N' para 'V=10'



GRÁFICA 12: Tiempo frente a 'N' para 'V=12'



GRÁFICA 13: Tiempo frente a 'N' para 'V=14'



GRÁFICA 14: Tiempo frente a 'N' para 'V=16'

AI.3. Análisis en función del número de columnas ‘C’

AI.3.1. Comparativa por número de ejemplos ‘N’

Este apartado contiene 10 gráficas extraídas de los datos de la ‘vista 2’, una por cada valor de ‘N’ (una columna de datos). En cada gráfica se representan 9 series de datos que se corresponden con los segmentos verticales en que se divide cada columna de datos (9 bloques). El eje X representa el número de columnas ‘C’ del conjunto de datos procesado.

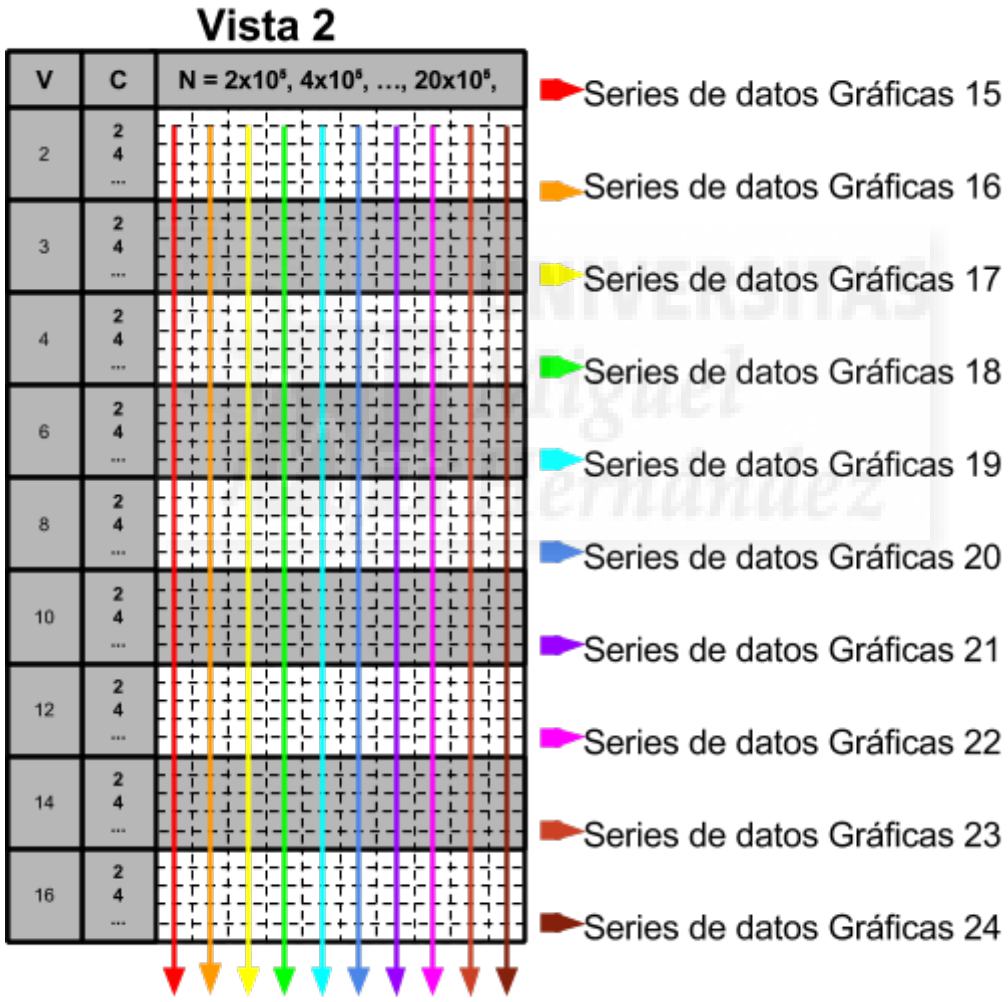
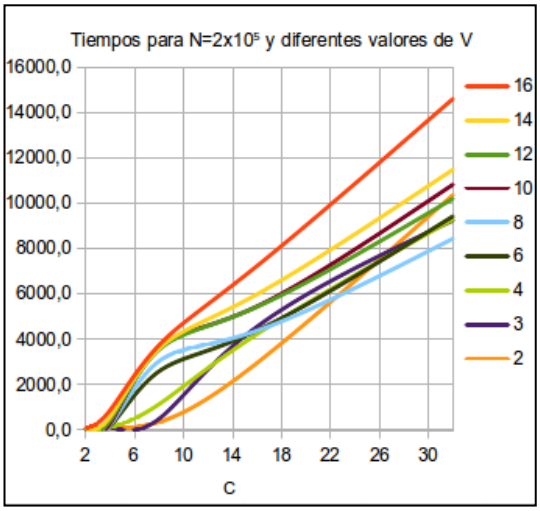
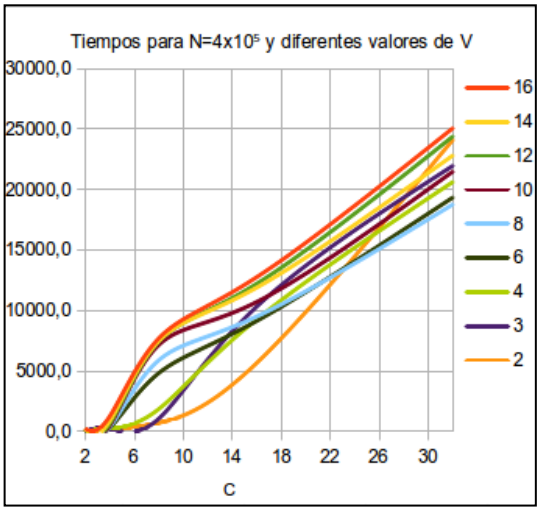


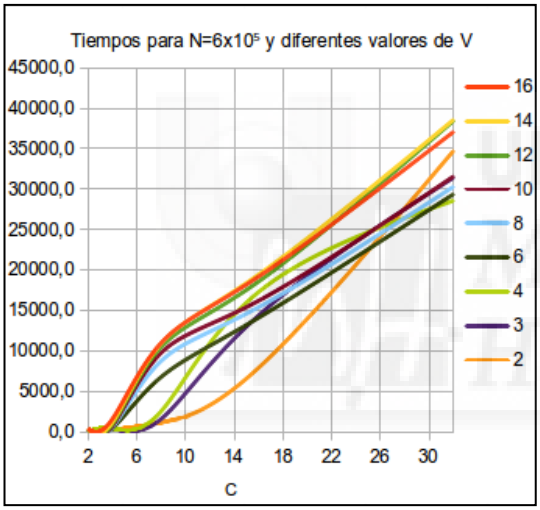
Figura AI.6: Origen de las series de datos de las gráficas del epígrafe AI.2.1



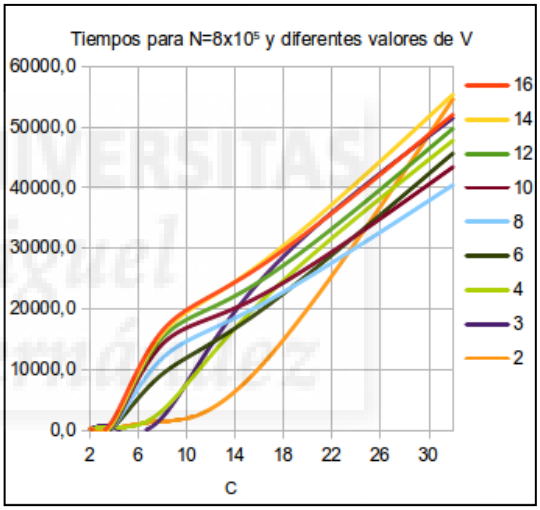
GRÁFICA 15: Tiempo frente a 'C' y 'N=2x10^{5},



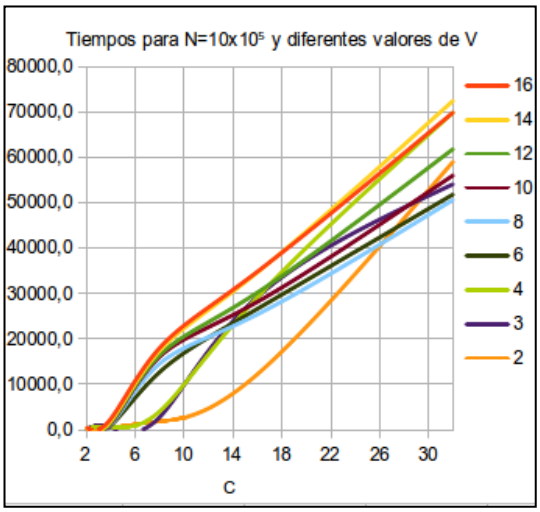
GRÁFICA 16: Tiempo frente a 'C' y 'N=4x10^{5},



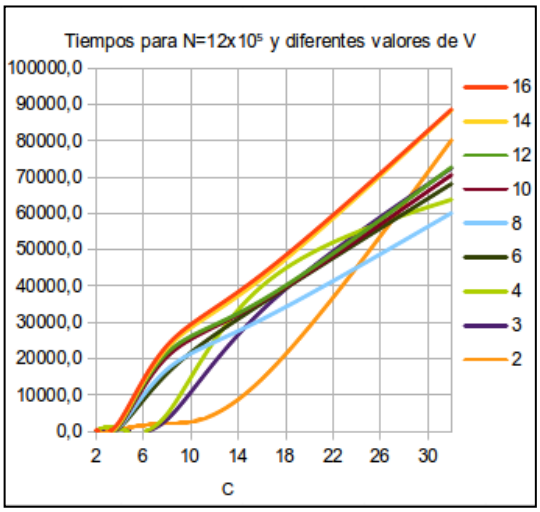
GRÁFICA 17: Tiempo frente a 'C' y 'N=6x10^{5},



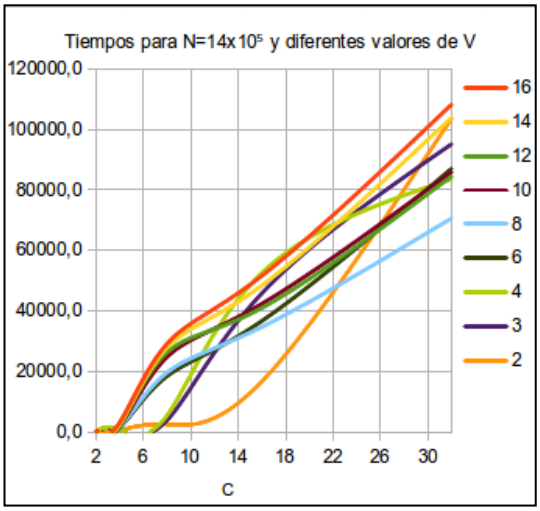
GRÁFICA 18: Tiempo frente a 'C' y 'N=8x10^{5},



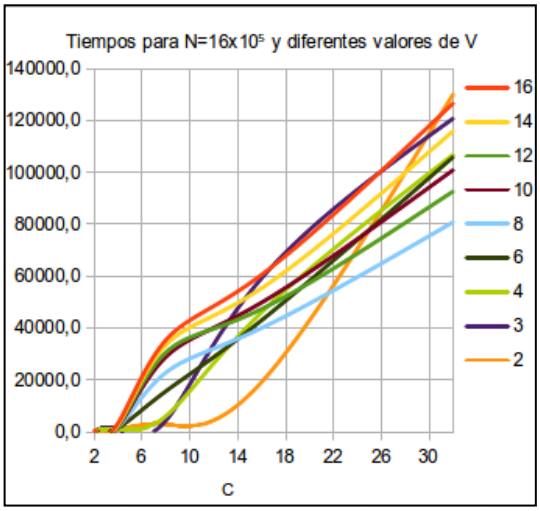
GRÁFICA 19: Tiempo frente a 'C' y 'N=10x10^{5},



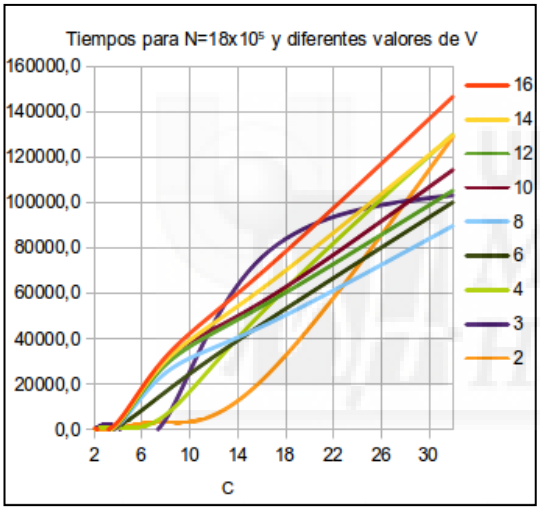
GRÁFICA 20: Tiempo frente a 'C' y 'N=12x10^{5},



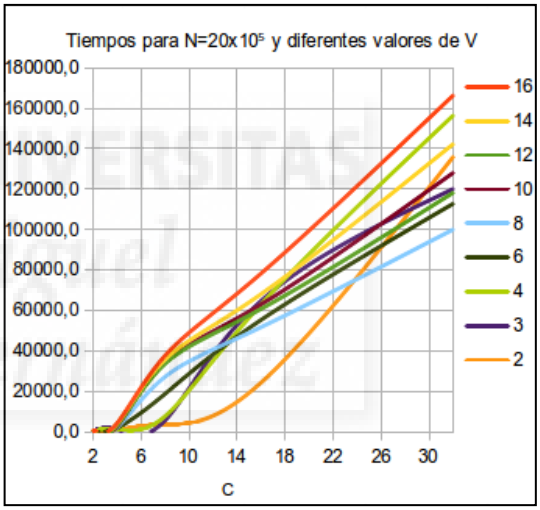
GRÁFICA 21: Tiempo frente a 'C' y 'N=14x10^5'



GRÁFICA 22: Tiempo frente a 'C' y 'N=16x10^5'



GRÁFICA 23: Tiempo frente a 'C' y 'N=18x10^5'



GRÁFICA 24: Tiempo frente a 'C' y 'N=20x10^5'

AI.3.2. Comparativa por número de valores ‘V’

Este apartado contiene 9 gráficas extraídas de los datos de la ‘vista 2’, una por cada valor de ‘V’ (9 bloques). En cada gráfica se representan 10 series de datos que se corresponden con las 10 columnas de datos (valores de ‘N’) de cada bloque. El eje X representa el número de columnas ‘C’ del conjunto de datos procesado.

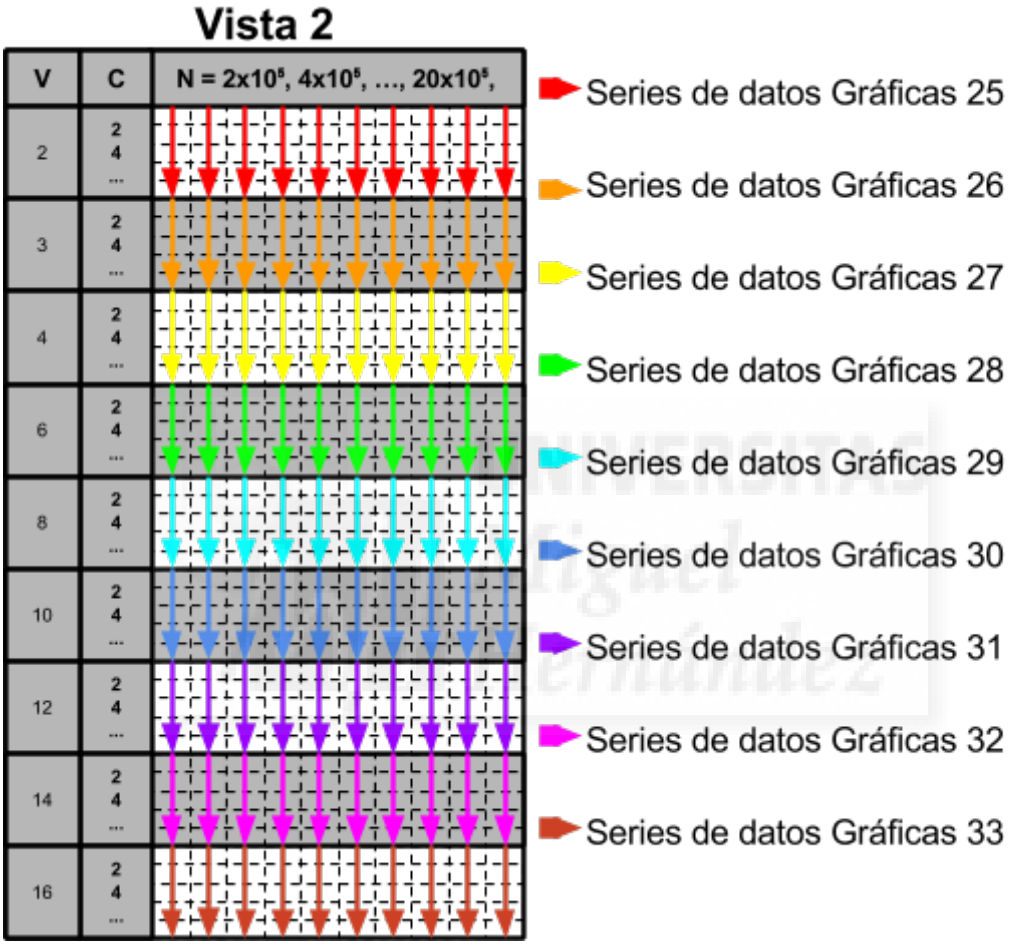
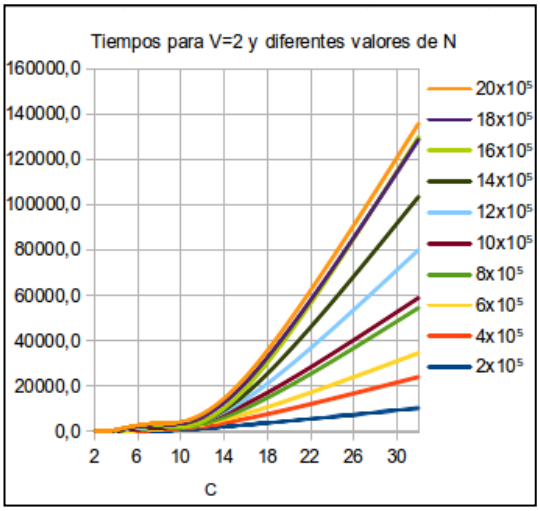
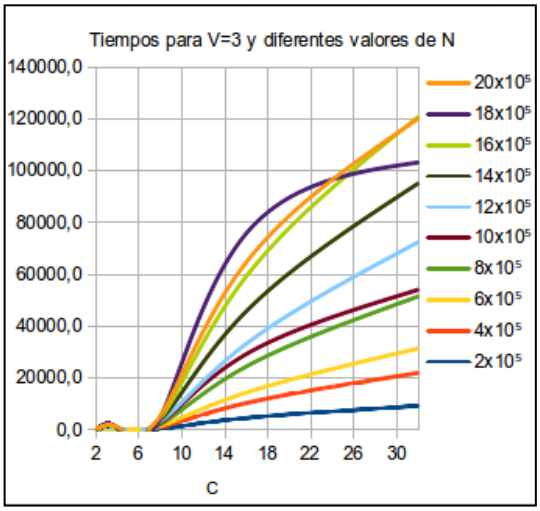


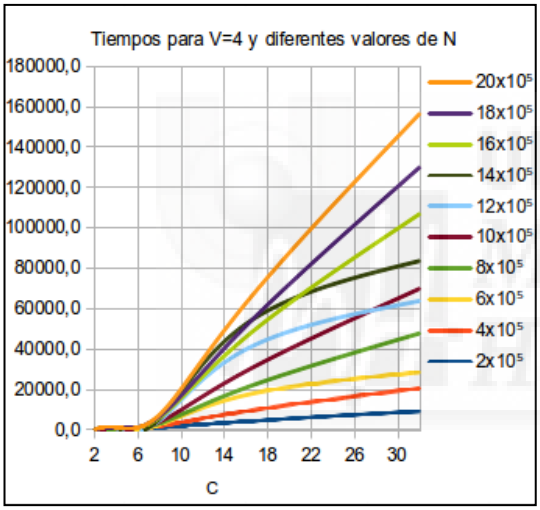
Figura AI.7: Origen de las series de datos de las gráficas del epígrafe AI.2.1



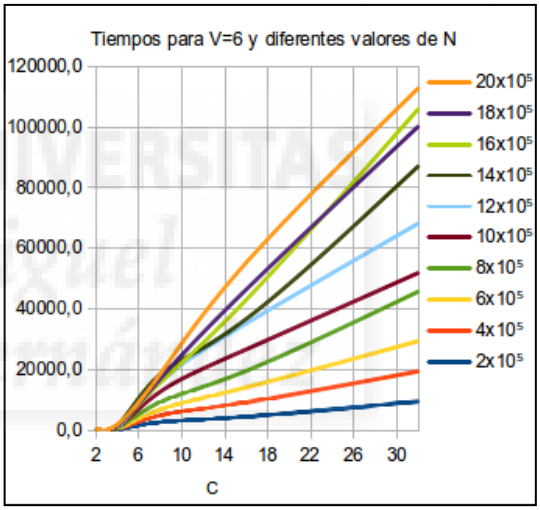
GRÁFICA 25: Tiempo frente a 'C' para 'V=2'



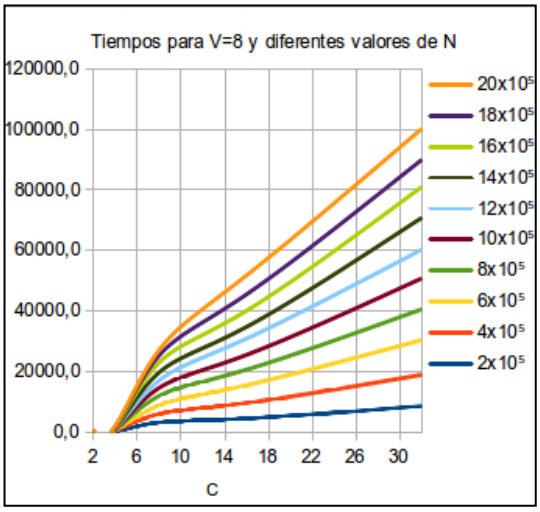
GRÁFICA 26: Tiempo frente a 'C' para 'V=3'



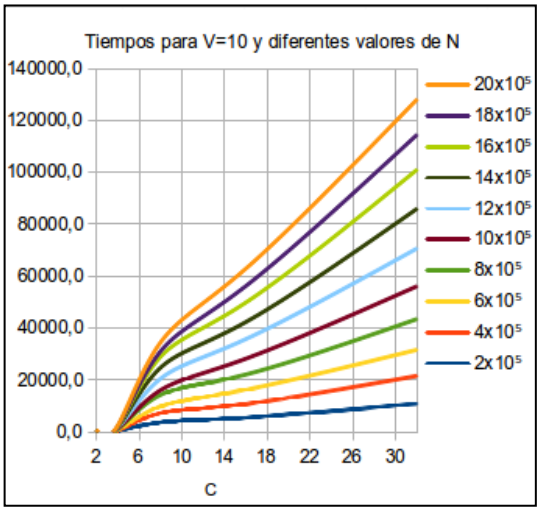
GRÁFICA 27: Tiempo frente a 'C' para 'V=4'



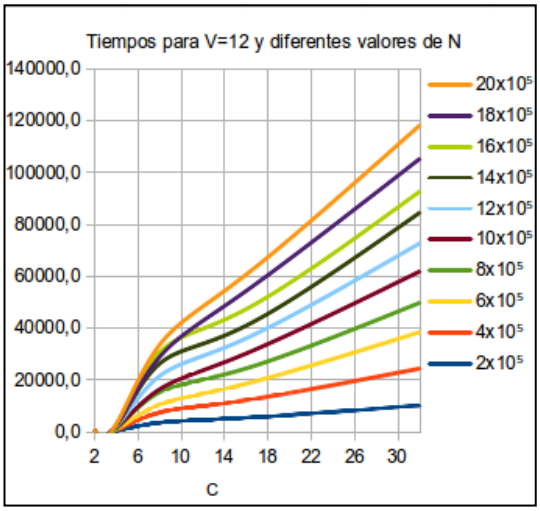
GRÁFICA 28: Tiempo frente a 'C' para 'V=6'



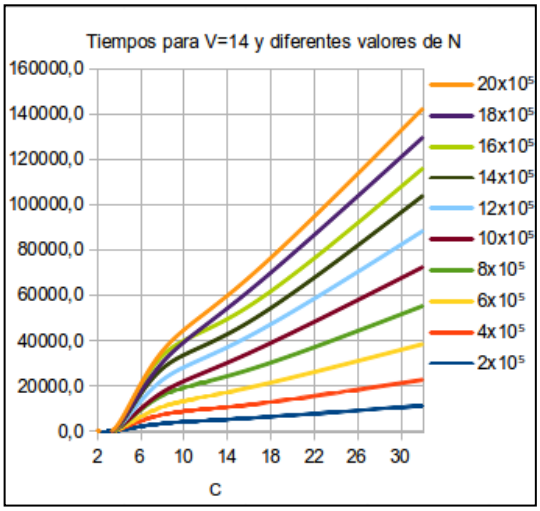
GRÁFICA 29: Tiempo frente a 'C' para 'V=8'



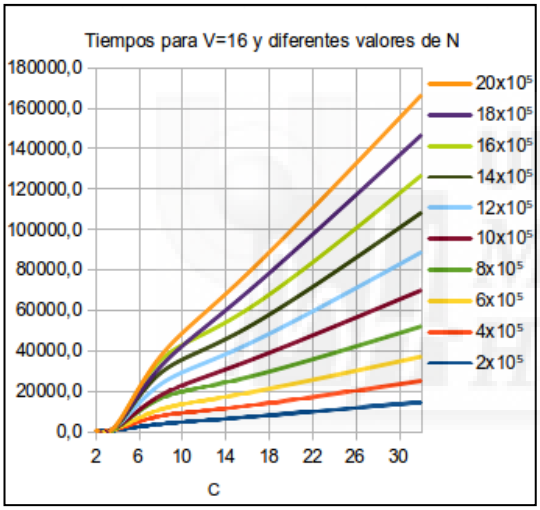
GRÁFICA 30: Tiempo frente a 'C' para 'V=10'



GRÁFICA 31: Tiempo frente a 'C' para 'V=12'



GRÁFICA 32: Tiempo frente a 'C' para 'V=14'



GRÁFICA 33: Tiempo frente a 'C' para 'V=16'

AI.4. Análisis en función del número de valores ‘V’

AI.4.1. Comparativa por número de ejemplos ‘N’

Este apartado contiene 10 gráficas extraídas de los datos de la ‘vista 1’, una por cada valor de ‘N’ (columnas de datos). En cada gráfica se representan 5 series de datos que se corresponden con los 5 segmentos verticales (bloques) en que se divide cada columna de datos (valores de ‘N’). El eje X representa el número de valores ‘V’ del conjunto de datos procesado.

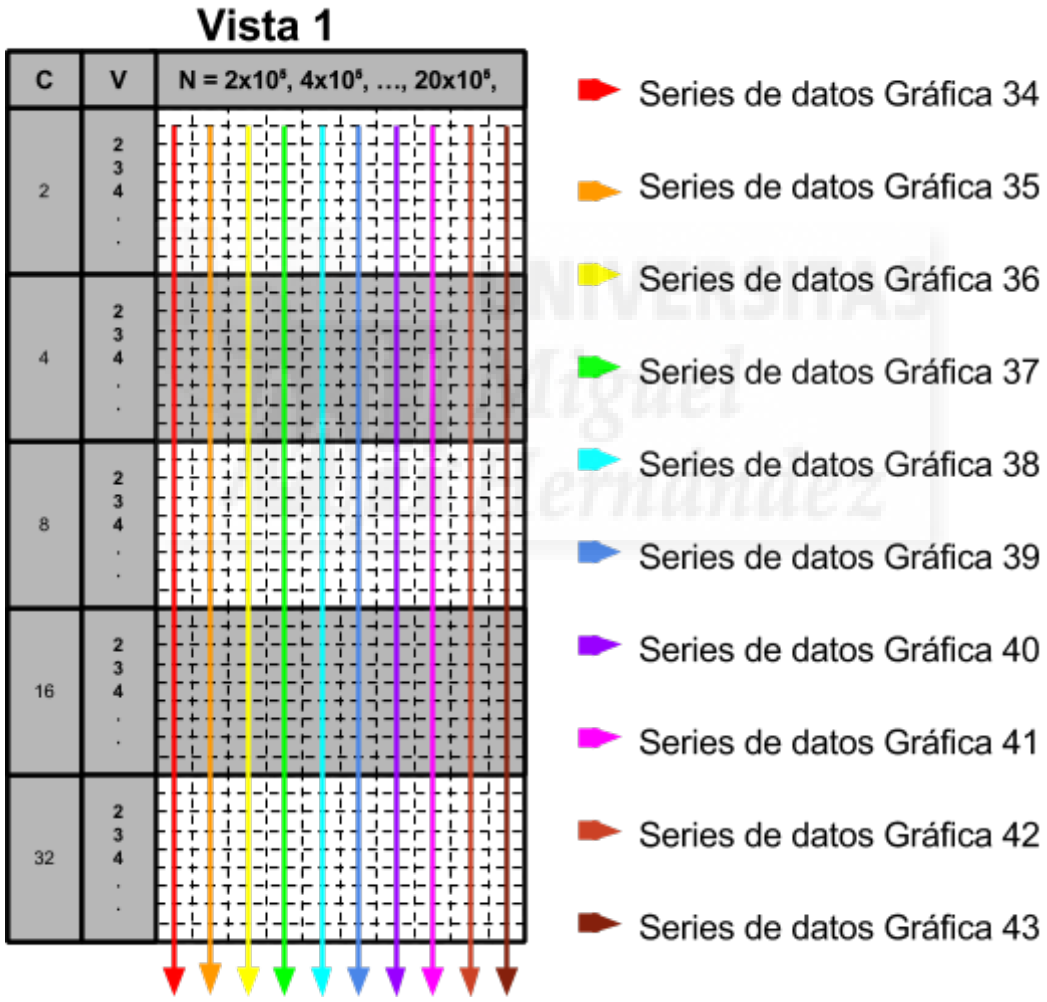
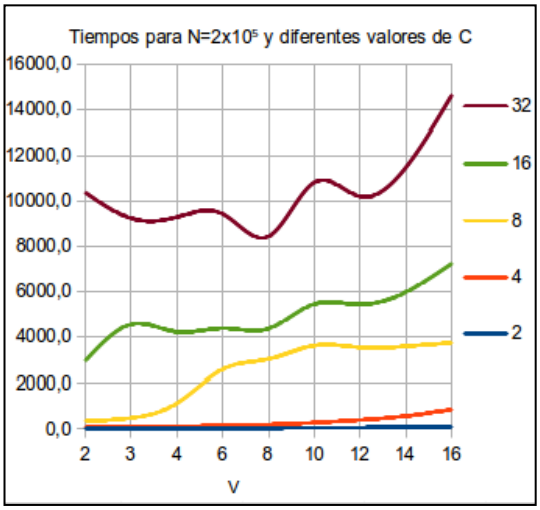
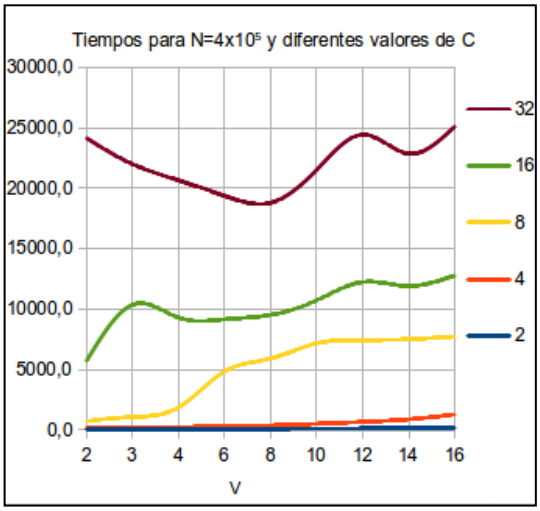


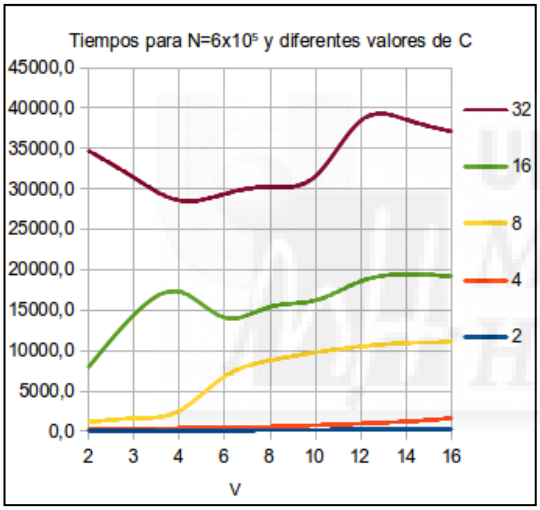
Figura AI.8: Origen de las series de datos de las gráficas del epígrafe AI.1.1



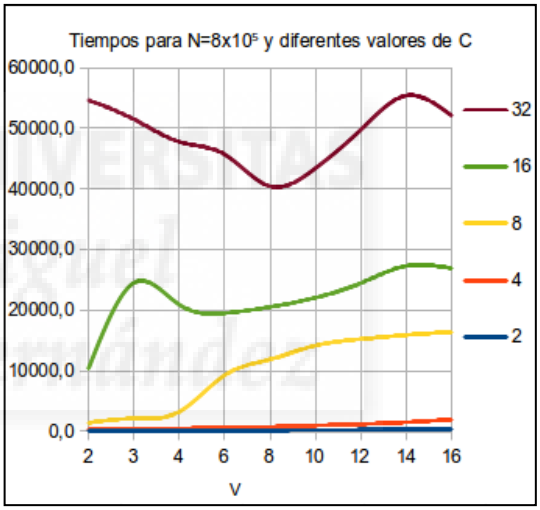
GRÁFICA 34: Tiempo frente a 'V' y 'N=2x10^{5},



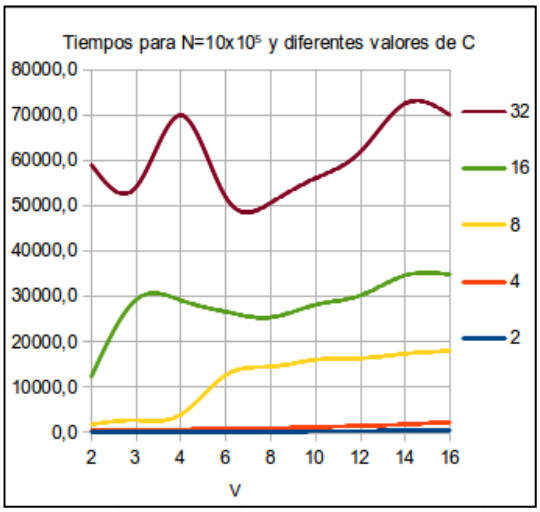
GRÁFICA 35: Tiempo frente a 'V' y 'N=4x10^{5},



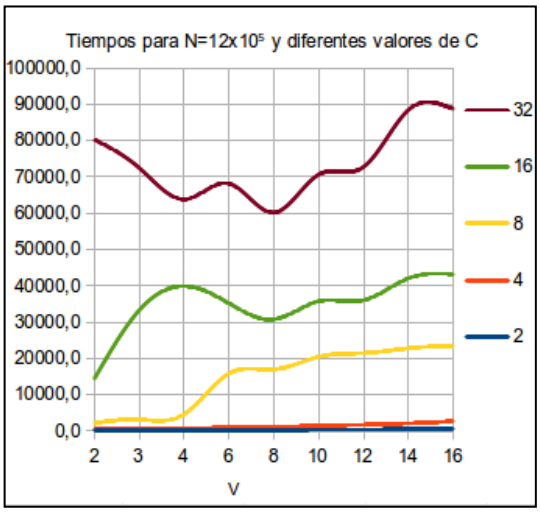
GRÁFICA 36: Tiempo frente a 'V' y 'N=6x105'



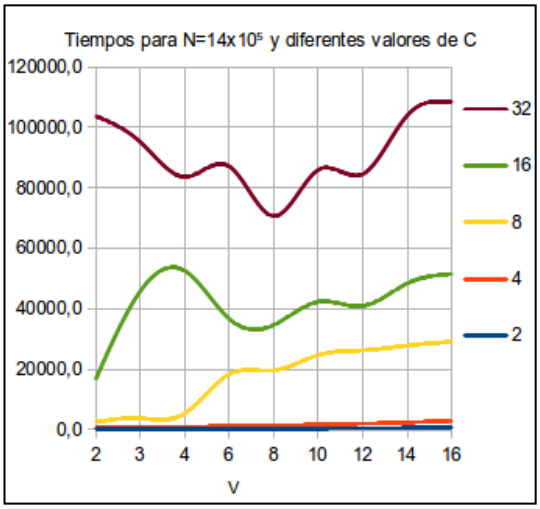
GRÁFICA 37: Tiempo frente a 'V' y 'N=8x105'



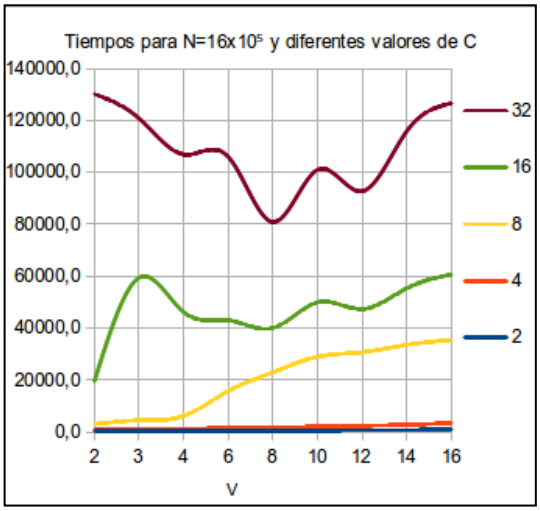
GRÁFICA 38: Tiempo frente a 'V' y 'N=10x10^{5},



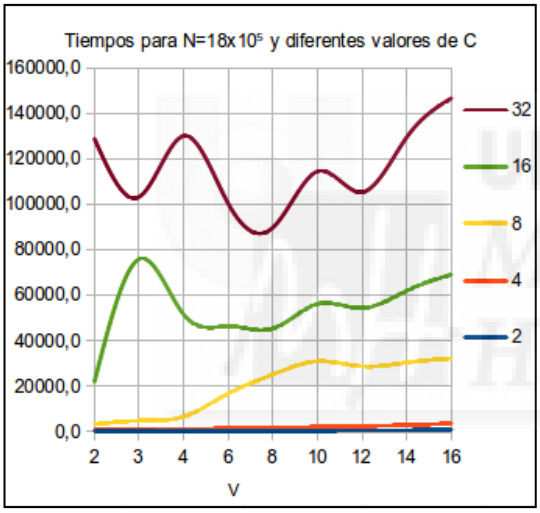
GRÁFICA 39: Tiempo frente a 'V' y 'N=12x10^{5},



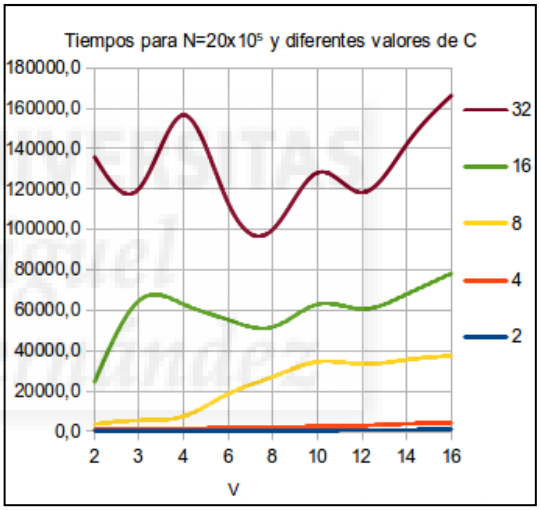
GRÁFICA 40: Tiempo frente a 'V' y 'N=14x10^5'



GRÁFICA 41: Tiempo frente a 'V' y 'N=16x10^5'



GRÁFICA 42: Tiempo frente a 'V' y 'N=18x10^5'



GRÁFICA 43: Tiempo frente a 'V' y 'N=20x10^5'

AI.4.2. Comparativa por número de columnas ‘C’

Este apartado contiene 5 gráficas extraídas de los datos de la ‘vista 2’, una por cada valor de ‘C’ (5 bloques). En cada gráfica se representan 10 series de datos que se corresponden con las 10 columnas de datos (valores de ‘N’) de cada bloque. El eje X representa el número de valores ‘V’ del conjunto de datos procesado.

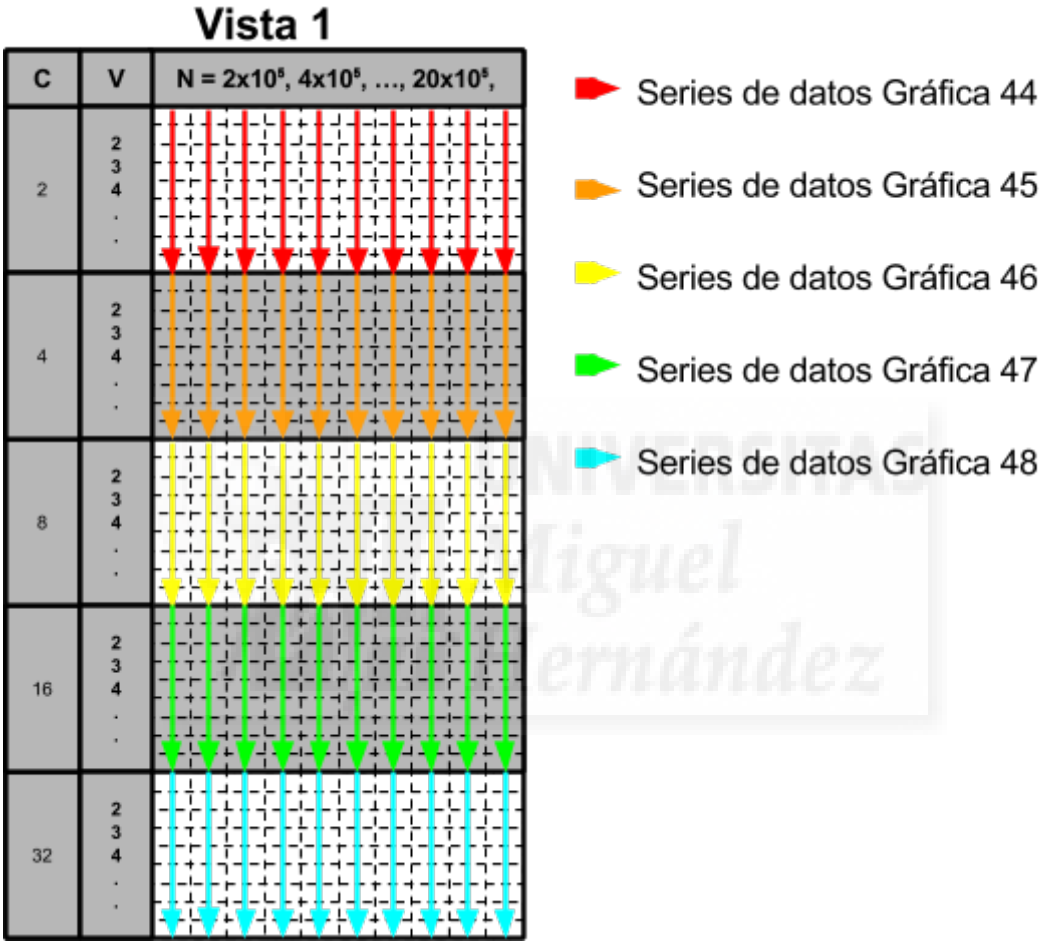
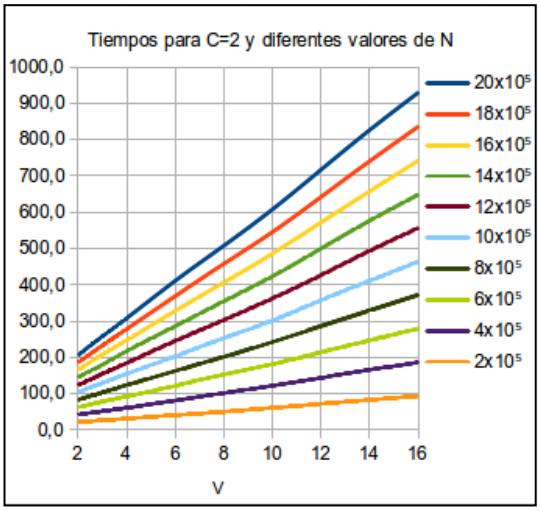
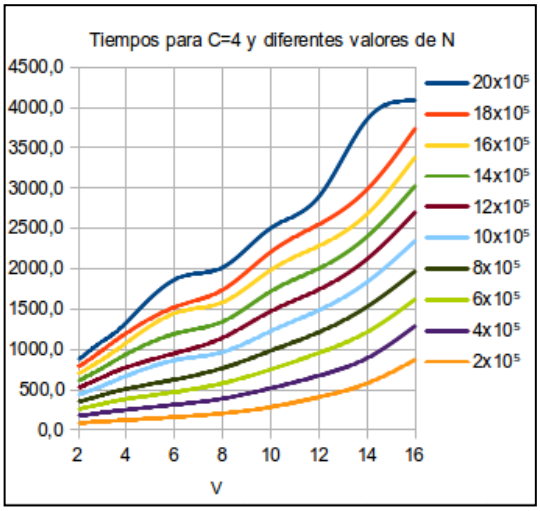


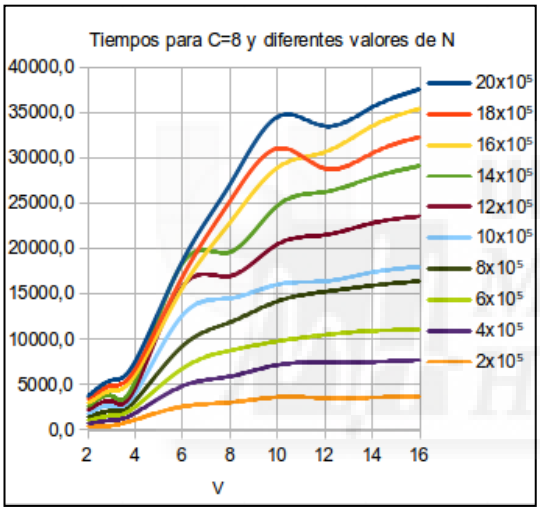
Figura AI.9: Origen de las series de datos de las gráficas del epígrafe AI.1.1



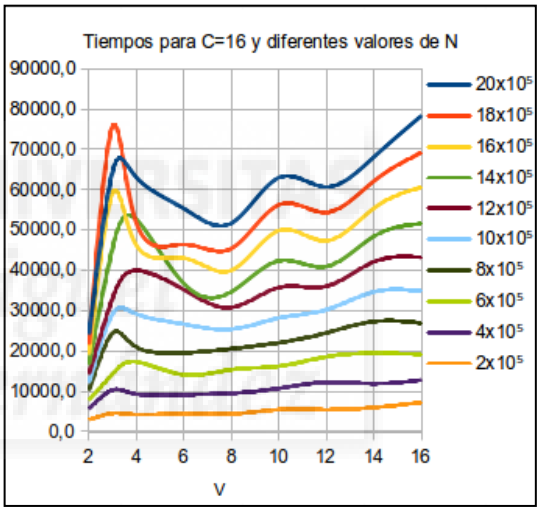
GRÁFICA 44: Tiempo frente a 'V' para 'C=2'



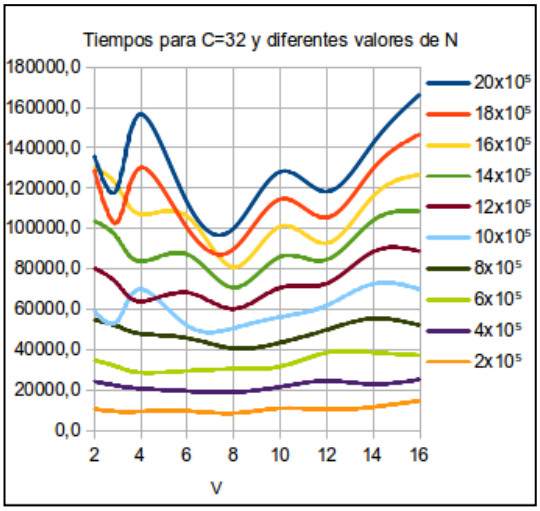
GRÁFICA 45: Tiempo frente a 'V' para 'C=4'



GRÁFICA 46: Tiempo frente a 'V' para 'C=8'



GRÁFICA 47: Tiempo frente a 'V' para 'C=16'



GRÁFICA 48: Tiempo frente a 'V' para 'C=32'

UNIVERSITAS
Miguel
Hernández

Anexo II
Verificación de la hipótesis del Objetivo 5

AII.1. Formulación de la hipótesis

En el objetivo 5 de la presente tesis (epígrafe 2.2) se plantea verificar la siguiente hipótesis: dados dos conjuntos de ejemplos, 'E_1' y 'E_2', caracterizados por los atributos 'N_1', 'C_1', y 'V_1' el primero y 'N_2', 'C_2' y 'V_2' el segundo, si se cumple la siguiente condición:

$$N_1 = N_2 \wedge \ V_1^{C1} = V_2^{C2}$$

Entonces se cumple que:

$$\boldsymbol{\Theta}(\ \mathrm{CREA}(E_1)\ ) \cong \boldsymbol{\Theta}(\ \mathrm{CREA}(E_2)\ )$$

Es decir, para un mismo número de ejemplos ('N') y un mismo valor 'V^C', los tiempos de procesamiento del algoritmo CREA deben coincidir o ser muy parecidos.

AII.2. Comprobación

Entre los conjuntos de datos utilizados de las pruebas empíricas, la tabla AII.1 indica, sombreados con el mismo color, aquellos conjuntos que cuyos valores 'V^C' coinciden. Además en la tabla AII.2 se muestra un extracto de la tabla AI.2 con los tiempos de ejecución del algoritmo CREA que van a permitir confirmar o refutar la hipótesis.

Tabla AII.1: Valores de V^C para los conjuntos de datos simulados

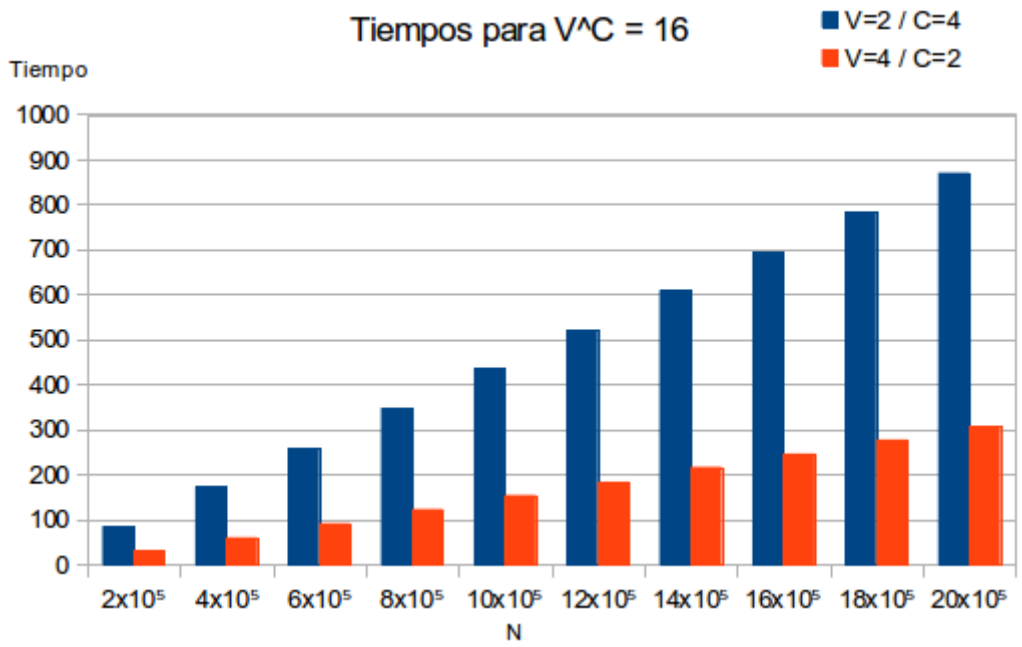
V^C	C = 2	C = 4	C = 8	C = 16	C = 32
V = 2	4	16	256	65536	4294967296
V = 3	9	81	6561	43046721	$1{,}85 \times 10^{15}$
V = 4	16	256	65536	4294967296	$1{,}84 \times 10^{19}$
V = 6	36	1296	1679616	$2{,}82 \times 10^{12}$	$7{,}96 \times 10^{24}$
V = 8	64	4096	16777216	$2{,}81 \times 10^{14}$	$7{,}92 \times 10^{28}$
V = 10	100	10000	100000000	$1{,}00 \times 10^{16}$	$1{,}00 \times 10^{32}$
V = 12	144	20736	429981696	$1{,}85 \times 10^{17}$	$3{,}42 \times 10^{34}$
V = 14	196	38416	1475789056	$2{,}18 \times 10^{18}$	$4{,}74 \times 10^{36}$
V = 16	256	65536	4294967296	$1{,}84 \times 10^{19}$	$3{,}40 \times 10^{38}$

Tabla AII.2: Tiempos de ejecución, resaltados según valor V^C

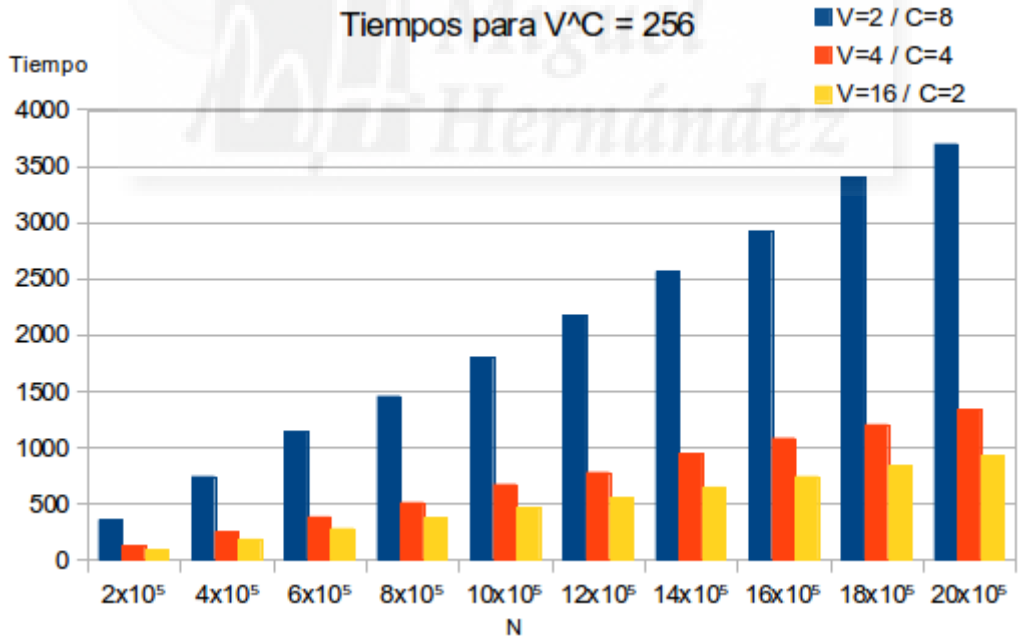
V	C	N= $2x10^5$	N= $4x10^5$	N= $6x10^5$	N= $8x10^5$	N= $10x10^5$	N= $12x10^5$	N= $14x10^5$	N= $16x10^5$	N= $18x10^5$	N= $20x10^5$
2	2	21,9	42,2	62,5	82,8	104,7	123,4	145,3	165,7	185,9	206,2
2	4	85,9	173,4	259,4	348,4	435,9	521,9	609,4	693,7	782,8	870,3
2	8	356,2	742,2	1143,8	1454,7	1803,1	2175,0	2559,4	2921,8	3400,0	3693,7
2	16	2961,0	5690,6	7904,7	10278,1	12220,3	14364,1	16687,5	19207,8	21746,8	24231,3
2	32	10381,2	24142,2	34725,0	54681,2	59109,4	80293,8	103779,7	130282,8	129020,3	136096,9
4	2	31,2	60,9	92,2	123,5	154,7	184,4	217,2	246,9	278,1	307,8
4	4	123,4	251,5	384,3	510,9	671,9	778,1	940,7	1082,8	1203,1	1334,3
4	8	1139,1	1871,9	2548,4	3243,8	3995,3	4635,9	5386,0	6084,4	6714,0	7543,8
4	16	4231,3	9300,0	17284,4	20937,5	29150,0	39946,9	52575,0	46112,5	51581,3	62971,9
4	32	9296,9	20646,9	28576,5	47801,6	69915,7	63828,1	83643,7	106926,6	130068,7	156540,6
16	2	93,7	185,9	279,7	371,9	464,1	556,3	648,4	742,2	835,9	929,6
16	4	873,5	1292,2	1623,4	1971,9	2348,4	2704,7	3029,7	3385,9	3740,6	4081,3
16	8	3746,9	7729,7	11017,2	16425,0	18003,1	23561,0	29109,4	35428,2	32306,3	37560,9
16	16	7239,1	12782,8	19184,3	26917,2	34836,0	43226,5	51579,7	60587,5	69168,8	78279,7
16	32	14615,6	25098,4	37075,0	52078,2	69934,3	88725,0	108309,4	126728,1	146712,5	166485,9

V^C=16 V^C=256 V^C=65536 V^C=$4{,}29x10^9$ V^C=$1{,}84x10^{19}$

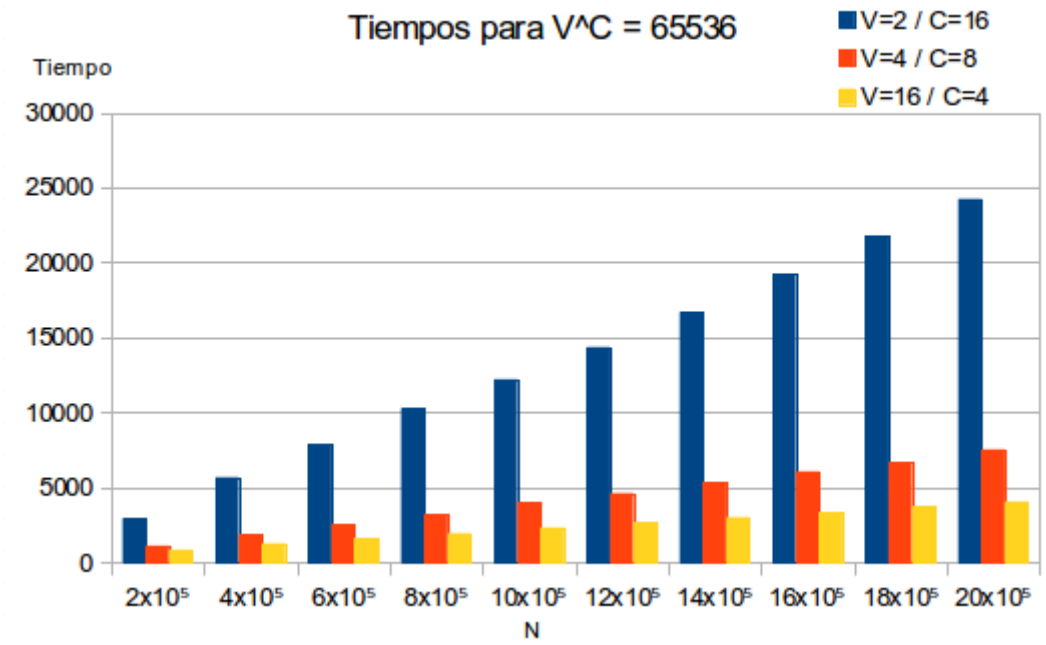
A la vista de los datos se puede concluir que la hipótesis planteada debe rechazarse ya que los tiempos no coinciden en ningún caso, para conjuntos con iguales valores de 'N' y de 'V^C', es más, la tendencia observada es que para valores de 'C' grandes los tiempos de procesamiento son mayores. Las gráficas 49 a 53 corroboran esta afirmación.



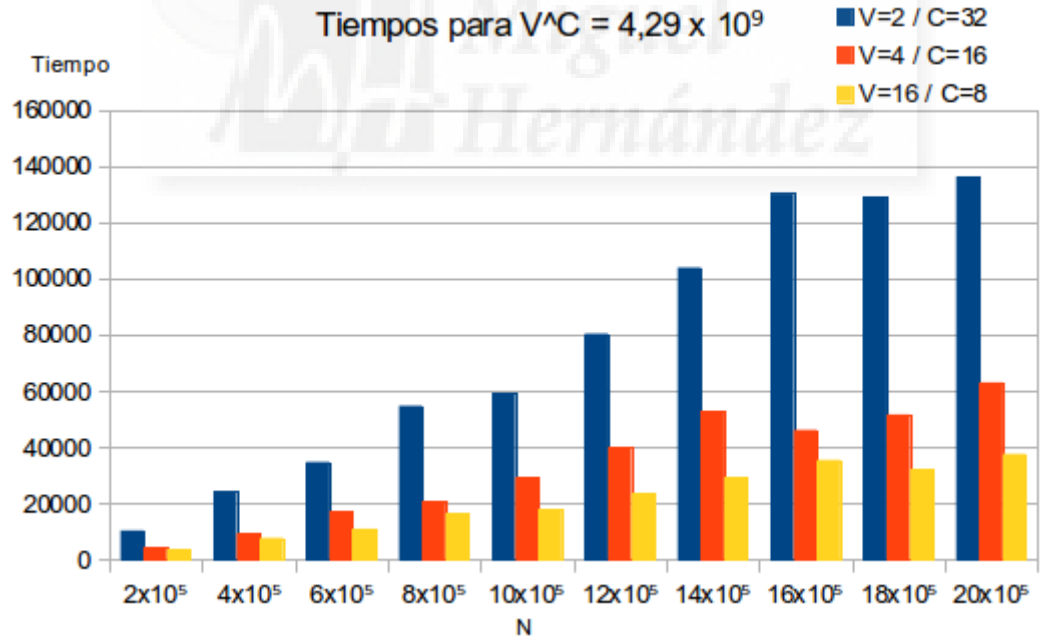
Gráfica 49: Comparativa de tiempos para conjuntos con V^C=16



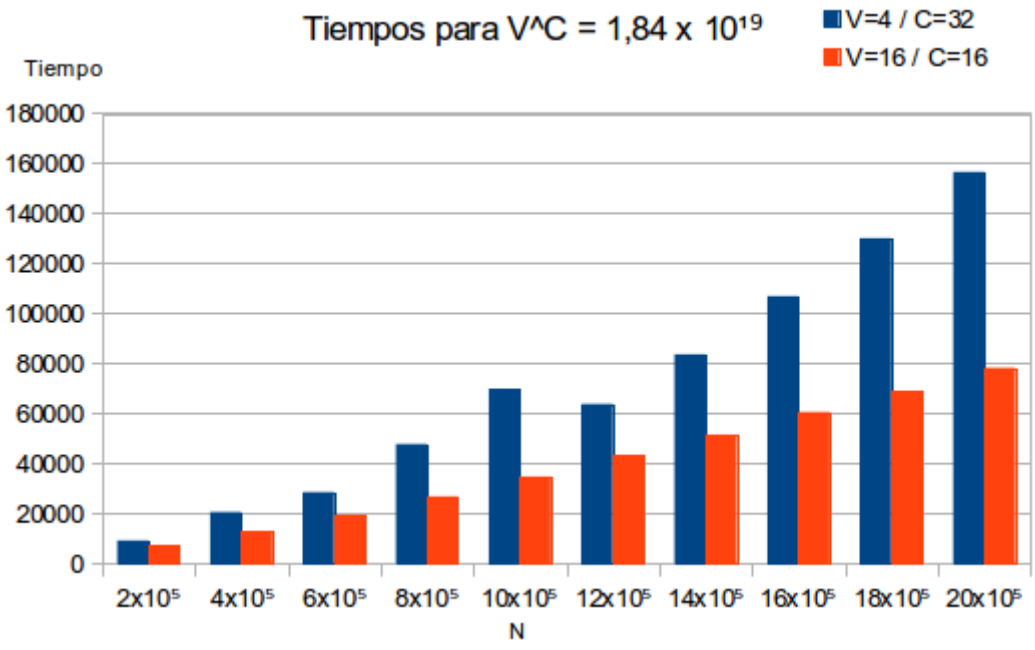
Gráfica 50: Comparativa de tiempos para conjuntos con V^C=256



Gráfica 51: Comparativa de tiempos para conjuntos con V^C=65536



Gráfica 52: Comparativa de tiempos para conjuntos con V^C=4,29 x 10^9



Gráfica 53: Comparativa de tiempos para conjuntos con V^C=1,84 x 10^{19}

UNIVERSITAS
Miguel
Hernández

Referencias bibliográficas

Ali, B., Mouakket, S. (2010).Integrating OLAP/SOLAP With E-Business: A New Conceptual Platform. Electronic-Business Intelligence: For corporate competitive advantages in the age of emerging technologies & globalization. Book Series: Advances in Intelligent Systems Research, 14, 240-246

Abu-halaweh, N.M., Harrison, R.W. (2009). Rule Set Reduction in Fuzzy Decision Trees. Annual meeting of the north american fuzzy information processing society, 237-240.

Aggarwal, N., Amit, K., Khatter, H., Aggarwal, V. (2012). Analysis the effect of data mining techniques on database. Advances in Engineering Software, 47, 164-169.

Almiñana, M., Escudero, L.F., Sanchez, C., Pérez, A., Santamaría, L., Rabasa, A., (2008). Reducting classification rules systems applied to thyroid functional diagnosis. International Biometric Conference, Dublin, 13 – 18 July 2008.

Almiñana, M., Rabasa, A., Santamaría, L., Escudero, L.F., Compañ, F.L., Pérez-Martín, A. (2010). Selecting the most accurate forecasting method for medical diagnosis.Breast cancer diagnosis - a case study.HEALTHINF conferente, Valencia, 20 - 23 January 2010.

Almiñana, M., Escudero, L.F., Pérez-Martín, A., Rabasa, A., Santamaría, L. (2012). A classification rule reduction algorithm based on significance domains. TOP (2012).

Awad, A., Goré, R., Hou, Z., Thomson, J., Weidlich, M. (2012).An iterative approach to synthesize business process templates from compliance rules. Information Systems 37, 714-736

Braaten, O. (1996). Artificial intelligence in pediatrics: Important clinical signs in newborn syndromes. Computers and biomedical research, 29, 3, 153-161 (Jun 1996).

Brassard, G., Bratley, p. (2000). Fundamentos de Algoritmia (spanish edition). Prentice Hall. ISBN: 848966000X.

Breiman, L., Friedman, J., Stone, C., Olshen, R.A. (1984). Classification and Regression Trees. Chapman & Hall/CRC (Jan 1984), ISBN: 0-412-04841-8.

Castellanos, M., Gupta, C., Wang, S., Dayal, U., Durazo, M. (2012). A platform for situational awareness in operational BI. Decision Support Systems 52, 869-883.

Chang, T.P., Chen, S.Y. (2012). An efficient algorithm of frequent XML query pattern mining for ebXML applications in e-commerce. Expert Systems with Applications, 39, 2183-2193.

Cheung, C.F., Li, F.L. (2012). A quantitative correlation coefficient mining method for business intelligence in small and medium enterprises of trading business. Expert Systems with Applications, 39, 6279-6291.

Chou, P.H., Li, P.H., Chen, K.K., Wu, M.J. (2010). Integrating web mining and neural network for personalized e-commerce automatic service. Expert Systems with Applications, 37, 2898-2910.

Dernoncourt, D., Hanczar, B., Zuckera, J.D. (2014). Analysis of feature selection stability on high dimension and small sample data. Computational Statistics and Data Analysis 71 (2014) 681–693.

Folino, F., Greco, G., Guzzo A., Pontieri L. (2011). Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction. Data & Knowledge Engineering 70 (2011) 1005–1029.

Gibson, M., Arnott, D., Jagielska, I., Melbourne, A. (2004).Evaluating the Intangible Benefits of Business Intelligence: Review & Research Agenda. International Conference on Decision Support Systems (DSS 2004): Decision Support in an Uncertain and Complex World. Proceedings, 295-305.

Goedertier, S., Weerdt, J.D., Martens, D., Vanthienen, J., Baesens, B. (2011). Process discovery in event logs: An application in the telecom industry. Applied Soft Computing, 11, 1697-1710.

Hall, P., Xue, J., (2014). On selecting interacting features from high-dimensional data. Computational Statistics and Data Analysis 71 (2014) 694–708.

Huang, Y., Lin, S. (1996). An efficient inductive learning method for object-oriented database using attribute entropy. IEEE Transactions on knowledge and data engineering, 8, 6, 946-951 (Dec 1996).

Huang, Z. Lu, X., Duan, H. (2011). Mining association rules to support resource allocation in business process management. Expert Systems with Applications, 38, 9483-9490.

Huang, T. C. (2012). Mining the change of customer behaviour in fuzzy time-interval sequential patterns.Applied Soft Computing, 12, 1068-1086.

Ismail, N.A., Husnayati, H. (2013). E-CRM Features in the Context of Airlines e-Ticket Jun M. (2012). Research of detecting e-business fraud based on Data Mining. International conference on computer science and network technology. 1-4, 2201-2204.

Purchasing: A Conceptual Framework.Proc.5th International Conference on Information and Communication Technology for the Muslim World (ICT4M).

Ke-Wu, Y., Jin-Fu, Z., Qiang, S. (2007). The application of ID3 algorithm in aviation marketing.IEEE International conference on grey systems and intelligent services, vols 1 and 2, 1284–1288.

Kuosa, T. (2011). Different approaches of pattern management and strategic intelligence. Technological Forecasting & Social Change, 78, 458-467.

Lee, S.L. (2010). Commodity recommendations of retail business based on decision tree induction. Expert Systems with Applications, 37, 3685-3694

Lee, S., Ryu, K., Shin, M., Cho, G.S. (2012). Function and service pattern analysis for facilitating the recon?guration of collaboration systems. Computers & Industrial Engineering, 62, 794-800.

Li, H., Sun, J., Wu, J. (2010). Predicting business failure using classi?cation and regression tree: An empirical comparison with popular classical statistical methods and top classification mining methods. Expert Systems with Applications, 37,895-5904.

Lin, F., Liang, D., Chen, E. (2011).Financial ratio selection for business crisis prediction. Expert Systems with Applications, 38, 15094-15102.

Lim, A.H., Lee, C.S. (2010). Processing online analytics with classification and association rule mining. Knowledge-Based Systems, 23, 248-255.

Liou, J.J., Tang, C.H., Yeh, W.C., Tsai, C.Y. (2011). A decision rules approach for improvement of airport service quality. Expert Systems with Applications, 38, 13723-13730.

Liu, B., Cao, S.G, He, W. (2011).Distributed data mining for e-business.Information technology & management. 12, 2, 67-79.

Lu, Y., Boukharouba, K., Boonaert, J., Fleury, A., Lecoeuche, S. (2014). Application of an incremental SVM algorithm for on-line human recognition from video surveillance using texture and color features. Neurocomputing 126, 132-140.

Luo, H., Chen, Y., Zhang, W, (2010). An improved ID3 algorithm based on attribute importance-weighted.International workshop on database technology and applications (proceedings 2010).

Luhn, H.P. (1958). A Business Intelligence System. IBM Journal. Oct 1958.

Martínez-Murcia, F.J., Górriz, J.M., Ramírez, J., Illán, I.A., Ortiz, A. (2014).Automatic detection of Parkinsonism using significance measures and component analysis in DaTSCAN imaging. Neurocomputing 126, 58-70

Pal, N.R., Chakraborty, S., Bagchi, A. (1997). RID3: An ID3-like algorithm for real data. Information Sciences 96, 3–4 (1997), 271–290.

Pan, W.T (2012). The use of genetic programming for the construction of a financial management model in an enterprise.Applied Intelligence, 36, 2, 271-279.

Peng, K., Xu, D.Y. (2011). Modeling of Potential Customers Identification Based on Correlation Analysis and Decision Tree. Lecture Notes in Computer Science Volume: 6677, 566-575.

Pugna, I.B., Albescu, F., Sova, R (2011). Beyong reporting in business intelligence: Intelligence through analytics. International conference accounting and management information systems, (proceedings 2011), 6th ed. 1110-1124.

Qu, Z.M., Liang, X.Y. (2009). Application of Grey Relational Clustering and Data Mining in Data Flow of E-Commerce. Proc. International Conference on Computational Intelligence and Natural Computing, Vol. 1

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples.In D. Michie (Ed.), Expert systems in the micro electronic age.Edinburgh University Press.

Quinlan, J.R. (1986). Induction of Decision Trees. Machine Learning 1: 81-106, 1986. Kluwer Academic Publishers, Boston.

Quinlan, J.R. (2006). Bagging, Boosting, And C4.5. University of Sydney.Technical Report.

Rabasa, A. (2009). Método para la reducción de Sistemas de Reglas de Clasificación por dominios de significancia. Tesis Doctoral, Universidad Miguel Hernández de Elche.

Rabasa, A., Molina, J. M., Puerto, H., Rodríguez-Sala, J.J., Santamaría,L., Ruiz-Canales, A. (2011). Identifying the most relevant issues to minimize automatized irrigation water supply in vineyards using Data Mining. Ecosystems and Sustainable Development VIII, 353-362.

Romdhane, L.B., Fadhel, N., Ayeb, B. (2010). An efficient approach for building customer profiles from business data. Expert Systems with Applications 37, 1573-1585.

Seng, J.L., Chen, T.C. (2010). An analytic approach to select data mining for business decision. Expert Systems with Applications, 37, 8042-8057.

Seng, J.L, Chiu, S.H. (2011). A generic construct based workload model for business intelligence benchmark. Expert Systems with Applications 38, 14460-14477.

Shannon, C.E. (1948). A Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October, 1948.

Sharma S., Osei-Bryson, K.M., Kasper, G.M.(2012). Evaluation of an integrated Knowledge Discovery and Data Mining process model. Expert Systems with Applications, 39, 11335-11348.

Shih, M.J., Liu, D.R, Hsu, M.L. (2010). Discovering competitive intelligence by mining changes in patent trends. Expert Systems with Applications, 37, 2882-2890.

Tan, P.N., Kumar, V., Srivastava, J,.(2004). Selecting right objective measure for association analysis. Inf Syst 29: 293-313.

Tsai, C.F. (2009). Feature selection in bankruptcy prediction. Knowledge-Based Systems 22, 120-127.

Tsai, C.F., Hsiao, Y.C. (2010). Combining multiple feature selection methods for stock prediction: Union,intersection, and multi-intersection approaches. Decision Support Systems 50, 258-269.

Tsai, C.F., Lu, Y.H., Yen, D.C. (2012). Determinants of intangible assets value: The data mining approach. Knowledge-Based Systems, 31, 67-77.

Wirth, N. (1978). Algorithms + Data Structures = Programs, Prentice Hall PTR Upper Saddle River, NJ, USA (ISBN: 0130224189).

Xu, M. Wang, J.L., Chen, T. (2006). Improved decision tree algorithm: ID3(+). Controls and Information Sciences (lecture notes, 2006), 345, 141-149.

Yang, C.C., Prasher, S.O., Enright, P., Madramootoo, C., Burgess, M., Goel, P.K., Callum, I. (2002).Application of decision tree technology for image classification using remote sensing data.Elsevier. Agricultural Systems 76, 1101–1117.

Zaragozí, B.M., Navarro, J.T., Ramón, A., Rodríguez-Sala, J.J. (2011) "A study of drivers for agricultural land abandonment using GIS and Data Mining techniques". Ecosystems and Sustainable Development VIII, 363-374.

Zhang, Y., Chi, Z., Wang, D. (2005). Decision tree's pruning algorithm based on deficient data sets. International Conference on Parallel and Distributed Computing, Applications and Technologies (proceedings 2005), 1030-1032.

Zhang, Y., Li, M., Wang, Y. (2008).Decision Tree Classification Algorithm and Research on Its Improvement.International conference on computer science & education (proceedings 2008), 1093-1096.

Zhang, X., Wang, Y., Feng, D. (2008 - BIS).Application of ID3 Algorithm in Customer Classification Based on CRM.Recent advance in statistics application and related areas, pts 1 and 2, 574-578.

Zhang, P. (2009). Application of Decision Tree Classification in Classification of Credit Card Customers.Advances in management of technology, 1, 531-533.

Zhang, C.,Wang, X.J., Peng, Z.H. (2011). Dimensions for OLAP on Multidimensional Text Databases. Lecture Notes in Computer Science Volume: 6988, 272-281

Zhao, J., Chang, Z.P. (2006). Neuro-fuzzy decision tree by fuzzy ID3 algorithm and its application to anti-dumping early-warning system.IEEE International Conference on Information Acquisition, Vols 1 and 2, (Proceedings 2006), 1300-1304.