Universidad Carlos III de Madrid

TESIS DOCTORAL

Predicción y selección de características, mediante análisis local de la fiabilidad, para el mercado de valores y su extensión a problemas de clasificación y regresión

Autor:
D. Ricardo Martín Manso

Directores:
Dra. D[a]. Inés M. Galván León
Dr. D. Ricardo Aler Mur

ESCUELA POLITÉCNICA SUPERIOR
Departamento de Informática

Leganés, mayo de 2017

Contacto:
Ricardo Martín Manso
gricardo.martin@gmail.com
http://orcid.org/0000-0001-8862-4894

TESIS DOCTORAL

Predicción y selección de características, mediante análisis local de fiabilidad, para el mercado de valores y su extensión a problemas de clasificación y regresión

Autor: D. Ricardo Martín Manso

Directores: Dra. D^{a}. Inés María Galván León y Dr. D. Ricardo Aler Mur

Firma del Tribunal Calificador:

Tribunal		Firma
Presidente:	Dr. D. Pedro Isasi Viñuelas	.....
Vocal:	Dra. D^{a}. Sonia Schulenburg	.....
Secretario:	Dr. D. David Camacho Fernández	.....

Calificación: .....

En Leganés, a 8 de septiembre de 2017

Quiero dedicar este trabajo a mi familia: mis padres, mi hermana y a los seres queridos que ya no están con nosotros, por el cariño, la educación y la guía que me han brindado toda la vida.

Agradecimientos

Agradecer a Inés y Ricardo, mis directores de tesis, por su trabajo, sus consejos y su paciencia en el desarrollo de esta tesis.

A mi pareja Clara agradecerle, muy especialmente, la paciencia y los sacrificios que ha hecho para propocionarme el ambiente de tranquilidad necesario para sacar adelante este reto.

A mis compañeros del doctorado: y, muy destacadamente, a Jesús y a Moisés, por su ayuda, su amistad y por los buenos ratos que hemos pasado juntos.

Al resto de mis amigos, por la comprensión que han tenido, especialmente estos últimos años, donde los he tenido un tanto abandonados.

A todos ellos: ¡gracias!

Resumen

Esta tesis se encuadra dentro del ámbito del Aprendizaje Automático, un área de la Inteligencia Artificial (IA). A lo largo de la misma, se han diseñado y validado experimentalmente, nuevas técnicas de selección de atributos y de clasificación.

La motivación para el desarrollo de dichas técnicas, se basa en el deseo de implementar herramientas adecuadas para tratar problemas de selección de atributos y de clasificación en un dominio de especial dificultad: el mercado de valores.

Se ha partido de la hipótesis de que los factores que dificultan la clasificación correcta de los datos son, a menudo, un ratio desfavorable entre información y ruido, una alta dimensionalidad, escasez de patrones y desbalanceo del número de patrones de cada clase. Una vez identificados dichos factores, se han diseñado técnicas robustas frente a éstos, concretamente un algoritmo de selección de atributos (con diferentes variantes) y un algoritmo de clasificación.

Estas técnicas se han validado sobre un exhaustivo conjunto de problemas generados artificialmente y en problemas reales del mercado de valores.

Por último, se ha explorado la posibilidad de utilizar las nuevas técnicas de selección de atributos propuestas en problemas convencionales. Para ello, se han validado sobre un conjunto de dominios reales de uso común en Aprendizaje Automático, tanto para clasificación como para regresión.

Palabras clave: selección de atributos, clasificación, mercado de valores, fiabilidad, ruido, desbalanceo de clases.

Abstract

This thesis belongs to Machine Learning, an area of Artificial Intelligence (AI). During its development, new techniques of attribute selection and classification have been designed and validated empirically.

The motivation for the development of these techniques is based on the desire to implement adequate tools to deal with feature selection and classification problems in an area of particular difficulty: the stock market.

Based on the hypothesis that the factors which make data classification difficult are, frequently, a low ratio between information and noise; high dimensionality, small training samples, and class imbalance.

Once these factors have been identified, robust techniques to deal with them were designed, specifically a feature selection algorithm (with different variants) and a classification algorithm.

These techniques have been validated over exhaustive synthetic data sets and stock market problems.

Finally, the possibility of using the new feature selection techniques were explored in conventional problems. To this end, they were validated using a data set of actual domains, both for classification and regression.

Key words: feature selection, classification, stock market, reliability, noise, class imbalance, machine learning.

Índice general

Índice de figuras

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

Índice de tablas

1

Introducción

La idea original de esta tesis surgió por el deseo de resolver, adecuadamente, los problemas que se plantean al intentar predecir los movimientos futuros del mercado de valores a partir de los datos históricos del mismo.

Una revisión bibliográfica preliminar (ver Sección 2.1) hace pensar que la información contenida en los datos históricos del mercado de valores, en caso de existir, no parece ser muy relevante por lo que podría considerarse dicho mercado, básicamente, un sistema estocástico. Los estudios y teorías de varios autores parecen confirmar la idea de la poca cantidad de información utilizable en dichos datos, si bien, algunos estudios consiguen extraer cierta utilidad de los mismos.

En la misma línea, diferentes publicaciones parecen confirmar que el uso de técnicas tradicionales de aprendizaje automático sobre datos históricos del mercado de valores, han reportado resultados prometedores, si bien, pocos han obtenido resultados positivos de forma consistente para largos períodos de tiempo.

Llegados a ese punto, se plantea la siguiente pregunta: ¿No se encuentra información en los datos históricos porque no existe, o porque las técnicas utilizadas no son apropiadas para ello?

A partir de estos trabajos previos se han avanzado las siguientes hipótesis:

- Si bien la mayor parte del tiempo y para la mayoría de los valores cotizados, el movimiento de precios puede considerarse en gran medida aleatorio, es posible que, en momentos puntuales y para valores concretos sí pueda existir cierta cantidad de información utilizable.

- El problema podría plantearse en términos dinámicos donde la información aprovechable o ineficacias del Mercado, tal como se nombra en la literatura, una vez detectadas sean de utilidad breve antes de 'autodestruirse' por conocimiento general.

- Las técnicas de aprendizaje automático al uso, ven degradado en cierta manera su rendimiento en este tipo de problemas por la presencia de algunos factores que aportan dificultad al aprendizaje.

Si se dispusiera de una técnica de aprendizaje automático capaz de resolver problemas de clasificación de patrones, en presencia de dichos factores, podría afrontarse con ciertas garantías el problema de la predicción del movimiento de precios en el mercado de valores.

1.1. Descripción del problema

Podemos plantear el problema a resolver como un problema de clasificación, donde cada patrón de datos se genere a partir de datos históricos y cuya clase de salida, C_1 ó C_0, indica la ocurrencia futura o no de un evento de nuestro interés. Específicamente, se ha experimentado tomando como evento a predecir la subida de la cotización de una acción, de un día para otro.

En este tipo de problema, concurren ciertos factores que dificultan el aprendizaje automático: el número de patrones informativos respecto al total de patrones es relativamente pequeño y existen, potencialmente, un gran número de atributos irrelevantes (dando lugar a un ratio pequeño de información *versus* ruido), escasez de patrones, alta dimensionalidad y desbalanceo de clases (los eventos más interesantes suelen ser poco frecuentes).

Las técnicas clásicas de clasificación han demostrado, durante años, su valía en multitud de problemas, sin embargo, la presencia de los factores mencionados puede hacer que su funcionamiento se muestre deficiente o subóptimo Weiss (1995); Fawcett and Provost (1997); Japkowicz and Stephen (2002).

Atendiendo a la relación información frente a ruido, presente en los datos de entrenamiento, podemos visualizar distintos tipos de problemas (ver Fig. 1.1).

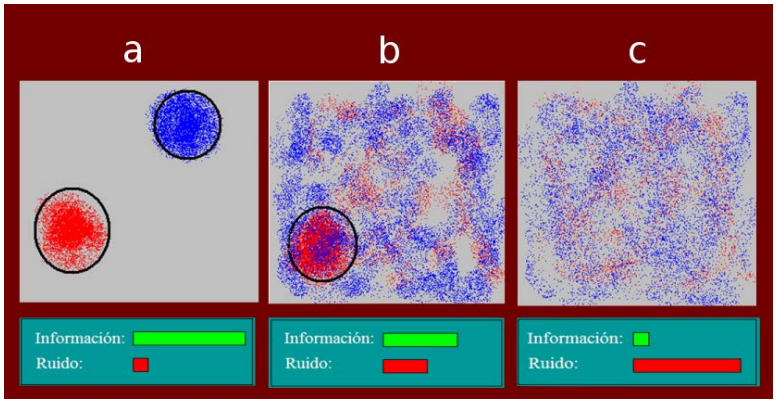

Figura 1.1: Información *vs* Ruido

Podemos observar cómo en los problemas similares al ejemplo **a** es fácil la separación entre los patrones de cada clase. En los problemas similares al ejemplo **b** ya es necesario hacer consideraciones probabilísticas y dependiendo de la sensibilidad y capacidades de la técnica utilizada podremos ser capaces de clasificar o no ciertos tipos de problemas. Por último, existen problemas como el ejemplo **c** en los que, o no existe información o ésta se encuentra más allá de los umbrales de detección de la técnica. En estos problemas es complicado determinar si una región del espacio de datos contiene información o la densidad observada de patrones de cada clase se debe únicamente a fluctuaciones producto del azar, especialmente si además concurre la circunstancia de escasez de patrones. El interés es esta tesis se centra en este tipo de problemas.

A la hora de desarrollar y configurar una técnica que sea capaz de tratar con estos problemas, se plantea un compromiso de diseño: si se estima demasiado alto el umbral de información confiable, ésto es, sobreestima las desviaciones producto del azar, gran parte de la información que podría utilizarse queda bajo el umbral y la técnica pierde sensibilidad. Por contra, si las desviaciones producto del azar se estiman por debajo de su valor real, encontraremos más información pero incurriremos en el riesgo de confundir información con simples efectos del azar. Con ello nuestra técnica perdería fiabilidad. Así, para poder detectar, con la máxima sensibilidad, toda la información aprovechable de un *conjunto de datos de entrenamiento*, será necesario que la técnica en cuestión sea capaz de cuantificar perfectamente las desviaciones producto del azar con el objetivo de que el umbral de detección pueda situarse lo más precisamente posible.

A lo largo de este documento de tesis, se usa a menudo la expresión ***Problemas Tipo-I***. Dicho término se emplea al único efecto de evitar enumerar, reiteradamente,

las características de ciertos problemas objeto principal de estudio en esta tesis. Cuando hablamos aquí de Problemas Tipo-I, nos referirnos a problemas que presentan características similares a las descritas anteriormente (en el ámbito del mercado de valores): relación extremadamente baja de información *versus* ruido, escasez de patrones, presencia de ruido, alta dimensionalidad y desbalanceo de clases. En cualquier caso, no se pretende con esto establecer una clasificación formal de problemas. Por otro lado, siendo que los factores aquí considerados pueden presentar todo un rango de valores continuos, la frontera de pertecencia de un problema dado a este subconjunto de problemas será difusa.

1.2. Objetivos

Para esta tesis, se plantean los siguientes objetivos:

- OBJETIVO-I: desarrollo de técnicas de aprendizaje automático, de selección de atributos y de clasificación, para abordar problemas de Tipo-I (mercado de valores).

  Para conseguir este objetivo, las técnicas a desarrollar tienen que atender a los siguientes requisitos:

  - Dado que se piensa que el sistema es dinámico, la información aparece y desaparece en breve, el número de patrones utilizados para entrenar nuestra técnica está limitado ya que, cuantos más patrones usemos para entrenamiento, más antigua u obsoleta estará la información aprendida. Este requisito nos lleva a la conclusión de que la técnica utilizada debe ser robusta frente a la escasez de patrones.
  - Si la relación información *versus* ruido en dichos problemas es muy baja, podemos pensar que la poca información extraible estará oculta entre muchos atributos irrelevantes que no aportarán información confiable al aprendizaje. Este requisito nos lleva a que la nueva técnica debe ser robusta a la presencia de muchos atributos irrelevantes, a una muy baja relación información/ruido y, con el fin de hacer el entrenamiento eficaz en el tiempo, debe ser capaz de enfrentar problemas de alta dimensionalidad.

- Dado que los eventos más interesantes de predecir, tanto desde el punto de vista estadístico[1] como del económico, son los que se presentan con poca frecuencia, la nueva técnica deberá trabajar cómodamente con clases desbalanceadas (el número de patrones de una clase puede ser muy distinto a los de la otra) y enfocar sus capacidades a clasificar correctamente la clase minoritaria. Por ejemplo, si quisieramos predecir el evento de que una cotización variara más de un 6 % de un día a otro, podríamos encontrarnos con un 3 % de patrones de clase C_1 frente a un 97 % de clase C_0.
- Dado que el problema presenta una gran incertidumbre, puede ser conveniente que la técnica incorpore un estado *Indeterminado* a su salida. Generalmente, las técnicas de clasificación asignan a cada patrón de datos una categoría/clase de entre las posibles para un problema dado. Sin embargo, en problemas de Tipo-I, es muy frecuente que tengamos que lidiar con patrones no clasificables o indeterminados. Es decir, es muy posible la presencia de patrones donde no exista información suficiente en los datos de entrada para clasificar de forma confiable, en algunas de las clases del problema. Así, una técnica que contemple dicho estado o clase de salida, puede suponer una especialización conveniente para tratar con este tipo de problemas.
- En cualquier caso, la nueva técnica debe poder establecer umbrales de confiabilidad de la solución, de forma que se pueda establecer el valor de confianza que se desea utilizar en la clasificación.

En estos problemas existen muy pocos patrones informativos (es decir, que realmente representan un suceso significativo y no una desviación producto del azar) y, de éstos, muchos lo son sólo parcialmente. Por ello, gracias al proceso de aprendizaje automático podremos recuperar algo de información pero no por ello resulta factible que podamos clasificar el 100 % de los patrones, dado que la gran mayoría de los mismos obedecerán sólo al azar. En este tipo de escenarios, las **medidas de bondad** de clasificación que observen el total de patrones, no parecen ser muy útiles dado que un porcentaje muy grande de los mismos sencillamente serán no clasificables.

[1] Según los postulados de la teoría de Información de Shannon, los patrones de la clase minoritaria (la menos frecuente) nos aportan muchísima más información que los de la mayoritaría.

Así, se establece como objetivo principal de la nueva técnica de clasificación encontrar el subconjunto de atributos y la región determinada por los valores que toman estos, donde se maximice la densidad de los patrones de la clase minoritaria respecto al total de patrones descartando, en lo posible, los efectos del azar. Formalmente, la medida que se desea maximizar es la *precision* en patrones C_1.

- OBJETIVO-II: dado que en muchos problemas de selección de atributos, tanto de clasificación como de regresión, pueden estar presentes algunos de los factores descritos anteriormente, el segundo objetivo que se plantea en esta tesis es estudiar la utilidad de las técnicas desarrolladas en problemas convencionales de clasificación y regresión.

  El estudio se va a circunscribir a las técnicas de selección de atributos, dado que parecen ser fácilmente generalizables. La técnica de clasificación no se va a intentar adaptar a problemas convencionales dado que se considera que por diseño está especializada en Problemas Tipo-I.

  Para los problemas de regresión se va a diseñar una técnica específica que discretice el problema de regresión y lo convierta en n problemas binarios de clasificación, de forma que pueda ser abordable con las mismas nuevas técnicas de selección.

  Para valorar la bondad del subconjunto de atributos, seleccionados por cada técnica, se hará uso de varias técnicas de clasificación y regresión que operarán sobre los atributos seleccionados. Los resultados obtenidos por dichas técnicas determinarán la valoración del subconjunto de atributos.

  Así, se plantean los siguientes objetivos:

  - Objetivo II.I: adaptación de las técnicas de selección de atributos desarrolladas, para problemas de clasificación.
  - Objetivo II.II: adaptación de las técnicas de selección de atributos desarrolladas, para problemas de regresión.

1.3. Estructura del documento

A continuación se describe la estructura y el contenido de este documento de tesis:

- En el presente capítulo, Introducción, se ha descrito el problema y se han precisado los objetivos a alcanzar.

- En el capítulo 2, Estado de la Cuestión, se hace una revisión de trabajos que abordan el problema de la predictibilidad del mercado de valores. A continuación, se hace una revisión de las técnicas actuales aplicables a selección de atributos. Por último, se describen brevemente las técnicas de selección de atributos y de clasificación utilizadas en la parte experimental de esta tesis.

- En el capítulo 3, Descripción de los métodos desarrollados, se describen las técnicas que se han desarrollado. Primeramente, se explican los fundamentos teóricos o conceptuales en los que se basan todas ellas. A continuación, se describen los algoritmos y las funciones de evaluación utilizadas. En estas últimas, se hará especial hincapié en la fiabilidad de las soluciones y su utilidad para enfrentar problemas que contengan un alto nivel de ruido e incertidumbre respecto a la cantidad de información. Se han desarrollado 4 variantes de un método de selección de atributos. LIA (Análisis Local de la Fiabilidad), LIA2 y LIA3 se presentan como nuevos métodos de selección de atributos, que implementan diferentes métodos de búsqueda de soluciones. LIAR es el nuevo método de selección de atributos desarrollado para utilizarse en problemas de regresión. Por último, se presenta también un nuevo método de clasificación (CLIA) que, siguiendo ideas similares a LIA, puede ser de utilidad específicamente en problemas Tipo-I.

- En el capítulo 4, Validación experimental en problemas de Tipo-I, se describen los experimentos realizados para comprobar el rendimiento de cada técnica desarrollada en este tipo de problemas. Primeramente, se explica cómo se han generado los problemas sintéticos. Posteriormente, se presentan los resultados sobre estos problemas, haciendo hincapié en el estudio del comportamiento de cada técnica frente a la presencia de ruido y la escasez de patrones. Los resultados obtenidos se comparan frente a los obtenidos con otras tres técnicas de selección de atributos: CFS-Subset, ReliefF y Chi-Squared.

A continuación, se plantea un experimento con datos reales del mercado de valores. Se detalla el procedimiento utilizado para el pretratamiento de los datos, la metodología utilizada y los resultados experimentales obtenidos con las técnicas propuestas. Finalmente, se presentan las conclusiones, tanto desde un punto de vista estadístico como desde la rentabilidad. Los resultados se presentan estableciendo comparativas frente a varias técnicas de inversión pasiva: *Comprar y Mantener* e Inversión Constante y, frente a los obtenidos con Random Forest utilizando Chi-Squared como selección de atributos.

- En el capítulo 5, Validación experimental en problemas de clasificación y regresión, se presentan los resultados obtenidos con los algoritmos de selección para problemas de clasificación y regresión. Para los problemas de clasificación se han empleado diferentes conjuntos de problemas reales, obtenidos de repositorios públicos: UCI y KEEL. Los resultados se han comparado con CFS-Subset, ReliefF y Chi-Squared en combinación con los clasificadores: J48, K-NN y MLP.

  Para los experimentos de regresión se han utilizado datos del repositorio KEEL y datos procedentes de una competición para la predicción de la producción fotovoltaíca. Los resultados obtenidos se han comparado con CFS-Subset y ReliefF. Como método de regresión se ha utilizado Máquinas de Vector Soporte y Gradient Boosting para el problema de radiación solar.

- En el capítulo 6, Conclusiones y trabajos futuros, se detallan las conclusiones obtenidas de este trabajo, se enumeran las aportaciones y publicaciones generadas y se apuntan algunas líneas futuras de ampliación del mismo.

2

Estado de la cuestión

En este capítulo se presenta el problema de la predictibilidad del mercado de valores analizando trabajos, tanto a favor como en contra, de la Hipótesis del Mercado Eficiente (HME). A continuación, se presentan publicaciones donde, haciendo uso de técnicas de Aprendizaje Automático, los autores intentan predecir ciertos eventos (habitualmente dirección del movimiento de un índice) en diferentes mercados.

Seguidamente se presentan algunos conceptos generales relativos al aprendizaje supervisado, haciendo especial incapié en dos factores que aportan dificultad al aprendizaje: la alta dimensionalidad y el desbalanceo de clases.

A continuación, se hace una revisión bibliográfica de técnicas de selección de atributos, entrando en mayor detalle en las que pueden ser de aplicación (*Filters*) en problemas similares al tratado en esta tesis.

Por último, se describen las técnicas de selección de atributos, de clasificación y de regresión utilizadas en la parte experimental de esta tesis, para establecer comparaciones frente a las técnicas propuestas.

2.1. El mercado de valores ¿Es predecible?

Los profesionales dedicados a la compra/venta de acciones con la finalidad de obtener ganancias generadas por la diferencia de precio de adquisición y el de venta, utilizan distintas técnicas con el fin de intentar predecir el comportamiento futuro del mercado. Dichas técnicas pueden agruparse en: 1. Análisis Fundamental (Graham and Dodd, 1934), análisis de los fundamentos económicos (tangibles) de cada empresa y 2. Análi-

sis Técnico (Kaufman, 2013; Schwager, 1996; Sewell, 2001), estudio de las cotizaciones históricas pasadas. Desde el ámbito académico/científico, se ha cuestionado y discutido sobre la eficacia real de dichos métodos (Malkiel, 1973a).

La crítica básica a la utilidad de dichas técnicas se fundamenta en la Hipótesis del mercado Eficiente (HME).

La idea de mercado eficiente está asociada al concepto de igualdad de condiciones, postulado por primera vez por Cardano en el año 1565, como un principio fundamental de los juegos de azar. Durante todo el siglo XX, (Malkiel, 1992; Fama, 1991; Malkiel and Fama, 1970; Mandelbrot, 1966; Samuelson, 1965; Fama, 1965; Bachelier, 1900) fueron dando forma a la definición moderna de mercado eficiente. Se dice así, que un mercado es eficiente cuando se cuenta con la suficiente liquidez y racionalidad económica por parte de los agentes, como para que cualquier tipo de información relevante sea absorbida por los precios de forma instantánea, generando un comportamiento aleatorio en ellos, lo que hace imposible su pronóstico sistemático.

En (Roberts, 1967) y principalmente, en (Fama, 1970) se plantea la existencia de tres niveles de eficiencia del mercado, atendiendo al grado de información que intervenga en la formación de los precios:

- Forma débil: la cotización de los títulos refleja toda la información histórica. No es posible utilizar estrategias basadas en el análisis de series históricas para lograr rendimientos que superen al propio mercado. La aparición de nuevas informaciones, nuevas noticias, desplaza aleatoriamente el precio arriba o abajo. Al aparecer éstas de forma impredecible en el tiempo, el cambio de los precios refleja también esta variación aleatoria. Esta hipótesis implica que el Análisis Técnico no aporta ninguna utilidad.

- Forma semifuerte: los actores del mercado cuentan con toda la información pasada y, también, con toda la información hecha pública acerca de un valor cotizado. Los precios se ajustan instantáneamente a toda información que se hace pública. La hipótesis semifuerte implica que el Análisis Fundamental no podrá lograr rendimientos superiores a los del mercado.

- Forma fuerte: los inversores, además de con la información pasada y con la información pública, cuentan con información privilegiada sobre los activos. Esta

hipótesis implica que el precio refleja toda la información posible y nadie puede obtener un rendimiento superior al del mercado.

A continuación, se van a enumerar las principales publicaciones, tanto a favor como en contra de la HME, ordenadas cronológicamente.

En el contexto de esta sección, se dice que un mercado es eficiente cuando los precios incorporan instantáneamente toda la información sobre un valor o un mercardo. Por contra, se dice que un mercado es ineficiente cuando los precios no reflejan toda la información o no lo hacen instantáneamente. Se dice, también, que existe una anomalía cuando los precios se comportan en un período concreto de tiempo de forma ineficiente.

La HME suele ir unida a la idea del llamado Paseo Aleatorio (Pearson, 1905) que viene a decir, de forma simplificada, que la serie de precios sigue una evolución aleatoria, donde los precios de un día D están en función del precio en D-1 más un diferencial aleatorio.

Trabajos que apoyan la HME:

(Osborne, 1959) concluyó que el logaritmo de los precios siguen una distribución aleatoria que se ajusta razonablemente bien a la calculada según la Teoría del movimiento Browtoniano.

(Alexander, 1961) publicó que el modelo de paseo aleatorio es el que mejor se ajusta a los datos pero encontró leptocurtosis en la distribución de los retornos.

(Fama, 1963; Mandelbrot, 1963) discuten la Hipótesis Estable Paretiana y concluyen que las cotizaciones (representadas aquí por los algoritmos naturales de los precios) parecen moverse aleatoriamente, siguiendo una distribución Estable Pareto-Levy.

(Fama, 1965) habla, por primera vez, del término Eficiencia de mercado, y concluye que el mercado evoluciona según un paseo aleatorio.

(Samuelson et al., 1965) expone la eficiencia de los mercados basando su explicación en procesos estocásticos de tipo *martingala*. La existencia de regularidades y/o tendencias no explica que pueda anticiparse si el siguiente movimiento será positivo o negativo.

(Mandelbrot, 1966) teoriza como en mercados competitivos, con inversores racionales, es de esperar retornos impredecibles y variación de precios según un modelo de martingala.

(Fama et al., 1969) aporta considerables evidencias de la eficiencia de los mercados.

En (Malkiel, 1973b) el autor defiende la HME, en contra de la valía del análisis técnico y del análisis fundamental. Entre otros argumentos, habla de la incapacidad de los expertos en estas disciplinas para batir de forma estable en el tiempo a los índices. También, aporta ideas empíricas sobre la autodestrucción de las anomalías una vez publicadas.

(Jackson, 1991) expone que las cotizaciones parecen reflejar toda la información más el coste de obtener la misma.

(Metcalf and Malkiel, 1994) presentan resultados que demuestran que los portfolios gestionados por expertos no consiguen superar al mercado.

(Seiler and Rom, 1997) analiza el NYSE entre los años 1885 y 1962 concluyendo que las cotizaciones diarias en dicho mercado son explicables con un modelo sencillo de Paseo Aleatorio. Su estudio respalda la validez de la forma semifuerte de la HME.

(Fama, 1998) considera que ninguna de las críticas a la HME, recibidas hasta el momento, justifican abandonar dicho modelo, al menos, para las inversiones a largo plazo.

(Malkiel, 2003) revisa y rebate las críticas recibidas por la HME en los años inmediatamente anteriores y concluye que la HME sigue siendo el modelo que mejor describe el comportamiento del mercado. No obstante, sí admite la existencia de anomalías durante breves períodos, no explicables únicamente por el modelo aleatorio.

(Schwert, 2003) muestra como según se van haciendo públicas anomalías del mercado, éstas son explotadas y, por ello, tienden a desaparecer. De esta forma, la publicación de anomalías o ineficiencias contribuye a que el mercado sea cada día más eficiente.

En (Timmermann and Granger, 2004) se vuelve a hablar de la autodestrucción de la predictibilidad en los mercados: una vez que una anomalía o ineficiencia se vuelve pública, se autodestruye cualquier utilidad que pudiera tener.

(Malkiel, 2005) estudia los resultados de fondos de inversión gestionados por profesionales y concluye que éstos intentan explotar ineficiencias del mercado, pero no son capaces de obtener mejores resultados que los índices que agrupan todas las acciones. Así, concluye, los índices reflejan instantáneamente toda la información disponible por lo que la HME sería un modelo correcto.

(Tóth and Kertész, 2006) publica evidencias de que el Mercado de Acciones de Nueva York es cada vez más eficiente.

(Wilson and Marashdeh, 2007) concluyen que el mercado es ineficiente a corto plazo y eficiente (conforme a la HME) a largo.

(Yen and Lee, 2008) realizan una revisión de las críticas actualidas a la HME, a fecha del artículo, y concluyen que la HME sigue siendo la mejor explicación posible a los movimientos del mercado.

En un par de trabajos (Marshall et al., 2008b,a), los autores analizan la utilidad de un amplio espectro de técnicas de análisis técnico (7.846 reglas), para operar en el Mercado Americano, tanto en operaciones de compra/venta de acciones como en *timing* (elección de punto de entrada y de salida) del mercado de futuros. En el primer estudio, se analizan los años 2002 a 2003, operando en el mercado de ETF, Exchange Traded Funds de AMEX. En el segundo estudio, se analizan 15 *commodities* para los años 1984 al 2005. En ninguno de los dos estudios se encuentra evidencia robusta de que el uso de reglas de análisis técnico presente una utilidad superior a la obtenible por varios *null models* (modelos que presentan algunas características estructurales similares a la serie de datos original pero siguen un patrón aleatorio) presentados por los autores. La metodología de valoración de reglas, seguida por los autores, está basada en (Bollerslev et al., 1992, 1994) donde se comparan los retornos obtenidos con cuatro modelos nulos: Paseo Aleatorio, AR(1), GARCH-M y E-GARCH.

(Sewell, 2011) publica una extensa relación de trabajos a favor y en contra de la HME. Apunta que uno de los problemas para evaluar si la HME es correcta consiste en que la hipótesis no está rigurosamente definida. El autor concluye con la idea de que si se valora la HME de una forma matemáticamente rigurosa se llega a la conclusión de que es falsa, pero si se relajan los criterios es totalmente cierta. Esto viene a significar que el mercado incorpora casi toda la información y lo hace casi instantáneamente.

Se puede concluir con respecto de los trabajos que apoyan la validez de la HME, con la idea de que es muy posible que la mayor parte del mercado, la mayor parte del tiempo, se mueva según un patrón aleatorio e impredecible. Esta idea está apoyada principalmente en: 1. La similitud entre el comportamiento de las cotizaciones y modelos generados de forma aleatoria, 2. Con la constación de la incapacidad de gestores profesionales de fondos para batir a los índices, de forma consistente en el tiempo y, 3. La utilidad efímera en el tiempo que podrían tener las ineficiencias encontradas en éste.

Cuestionamientos a la HME:

Las críticas a la HME se centran en demostrar la existencia de ineficiencias de mercado. Esto es, la existencia de estrategias que de forma estable en el tiempo permitieran obtener retornos mayores a los índices. En caso de existir estas estategias y de funcionar de manera consistente por largo tiempo, la HME se debería considerar refutada, al menos, en alguno de sus preceptos.

(Cootner, 1962; Osborne, 1962) concluyen, cada uno por su parte, que las cotizaciones del mercado de acciones no parecen seguir un paseo aleatorio. Los resultados apuntan a que los movimientos de las cotizaciones se concentran en rachas. En el mismo año (Moore, 1962) no encuentra auto correlación en series individuales de cotizaciones de diferentes empresas pero, sin embargo, sí lo encuentra en la serie del índice del mercado.

(Granger and Morgenstern, 1963) aplicando un análisis espectral a los precios de las acciones del Mercado de New York, concluyó que los movimientos de los precios a corto plazo son explicables por un paseo aleatorio pero, a largo plazo, los precios no parecen seguir dicho modelo.

Los premios Nobel de Economía del 2001, Arkelof, Spence, Stiglitz y Rothschild, en varios artículos (Akerlof, 1995; Rothschild and Stiglitz, 1976; Spence, 1973), cuestionan la hipótesis de mercado eficiente argumentando asimetrías en la información.

Por su parte, (Grossman and Stiglitz, 1980) presentan el modelo *noisy rational expectations model*: el ajuste de los precios, a toda la información disponible, no se hace de forma instantánea. Es decir, no todos los actores del sistema disponen de información de calidad en el mismo instante. Esto implicaría la existencia de oportunidades de predictibilidad en el corto plazo.

(Shiller, 1980) publica que los mercados muestran mayor volatilidad de precios que la explicable por la publicación de noticias económicas y pago de dividendos.

En (Grossman and Stiglitz, 1980), basándose en la idea de que la información tiene un coste, se postula que es imposible que los mercados sean perfectamente eficientes desde el punto de vista de la compartición de la información.

(Bondt and Thaler, 1985) muestra como los mercados sobrereaccionan (las variaciones de los precios se amplifican más allá de la pura racionalidad), siendo ésta una forma de ineficiencia de los precios.

(Fama and French, 1988) expone que puede calcularse autocorrelación negativa en las series de datos con horizontes de retorno de un año. En (Poterba and Summers, 1988) se llega a la misma conclusión: autocorrelación negativa a largo plazo y autocorrelación positiva, de menor entidad, a corto plazo.

(Chopra et al., 1992) concluye que los mercados sobrereaccionan.

(Jegadeesh and Titman, 1993) publicaron una sencilla estrategia, basada en comprar acciones de empresas ganadoras en el pasado y vender empresas perdedoras. Dicha estategia permite obtener retornos por encima del mercado. Es decir, la historia de una acción condiciona hasta cierto punto las probabilidades de movimientos futuros de ésta.

(Lakonishok et al., 1994) presenta evidencias de que una *estrategia value* (invertir en empresas con balances sólidos y generación de rendimientos económicos estables en el tiempo) permite obtener grandes rendimientos probablemente porque el comportamiento racional de muchos inversores es subóptimo.

Los estudios (Silber, 1994; Taylor and Allen, 1992; Taylor and Tari, 1989) calculan retornos del Análisis Técnico entre el 2 % y el 10 %, para los principales mercados, desde el año 1970 hasta primeros años de los 90.

(Haugen, 1995) presenta la sobrereacción del mercado en el corto plazo como causa de que funcionen las estrategias $Momentum$ (comprar activos fuertemente alcistas esperando que la tendencia continúe). Por contra, en el largo plazo el mercado tiende a corregir los excesos de la sobrereacción en el corto.

(Chan et al., 1996) estudió diferentes estrategias de $Momentum$ y publicó que los resultados sugieren que la nueva información se incorpora a los precios de manera gradual, no de forma instantánea.

(Balsara et al., 1996) propone flexibilizar y adaptar continuamente la parametrización de los indicadores técnicos utilizados, con el fin de adaptarse a las diferentes condiciones de un mercado evolutivamente cambiante.

(Andrew, 1997) cuestiona la validez del paseo aleatorio en los mercados de valores y presenta diferentes críticas a sus fundamentos, tanto a su versión original como a las variaciones posteriores (Movimiento Browtoniano y otros). Los autores confían en que una aplicación masiva de tecnología de computadores al análisis rápido de datos, el uso de instrumentos financieros que reduzcan el peso de las comisiones en la operativa y el avance de técnicas, como la psicología de masas, permita obtener retornos consistentemente superiores a los predichos por el paseo aleatorio y la HME.

2. ESTADO DE LA CUESTIÓN

En (Kaminsky and Schmukler, 1999) se presentan evidencias que apoyan el argumento que los agentes que operan en los mercados no son necesariamente racionales en el sentido económico y muchas veces toman decisiones basadas en elementos psicológicos, como el efecto manada o moda, guiados por el entusiasmo o la irritación.

(Exuberance, 2000) muestra que el movimiento de los precios históricamente no se justifica ni por las ganancias de las empresas ni por su reparto de dividendos.

En (Oh and Kim, 2002) se utilizan sistemas no lineales y modelos caóticos para intentar predecir el KOSPI 200 (Índice del Mercado Koreano) desde el año 1990 al 2000. Los autores concluyen que, haciendo uso de su método, obtienen rendimientos superiores al mercado de una forma estadísticamente significativa.

En (Sazuka, 2006; Tanaka-Yamawaki, 2003; Ohira et al., 2002) se estudia, utilizando métodos estadísticos, la no-aleatoriedad de los mercados financieros (ratio de cambio Dolar/Yen) en cortos intervalos de tiempo. De los tres estudios, se puede concluir que para períodos de tiempo cortos, menos de 5 minutos, la dinámica de dicho mercado no es completamente aleatoria. Además, las irregularidades probabilísticas encontradas presentan una larga persistencia en el tiempo (hasta 5 años).

(Stiglitz, 2010) presenta el comportamiento imprudente y descontrolado de Wall Street en los años previos al 2007, como evidencia de la irracionalidad del mercado.

En (Lee et al., 2010) se estudian 32 mercados de paises desarrollados y 32 de paises en desarrollo, en los años 1999 a 2007, y se concluye que los mercados de acciones no son eficientes.

(Gil et al., 2011) critica la hipótesis de eficiencia, en su forma fuerte, argumentando que los precios reflejan sólo de forma parcial la información disponible. En su estudio, sobre 10 mercados de valores internacionales, argumentan que díficilmente éstos se compartan de forma eficiente, para un valor de confianza del 95 %, en el período comprendido entre 2001 y 2010.

(Bastos and Caiado, 2011) presentan indicios de la relación entre el nivel de eficiencia y el de desarrollo, concluyendo que los mercados desarrollados son menos predecibles y más eficientes que los emergentes.

En (Lento and Gradojevic, 2011) se estudia el rendimiento de diferentes técnicas de Análisis Técnico sobre el *Dow Jones Industrial Index*, el *NASDAQ Composite Index* y el *Canada/US Exchange*, desde 1995 a 2004. Los resultados sugieren rendimientos mayores a la estrategia pasiva *Comprar y Mantener* . Sin embargo, este mayor rendimiento

no se reproduce de forma robusta en el tiempo, ni el diferencial es estadísticamente significativo para todos los indicadores.

En (Reboredo et al., 2012), utilizando intervalos de 5 a 60 minutos, con datos de *Standard and Poor's 500 Index*, comparan el rendimiento de varias técnicas estadísticas y de Aprendizaje Automático, en la predicción de los movimientos del índice, en términos económicos y estadísticos. Los autores concluyen que existen evidencias, aunque relativamente débiles, de predecibilidad de movimientos intradiarios.

En (Rechenthin and Street, 2013) se analiza la predictibilidad del *Standard and Poor's 500 Index*, en el año 2005, utilizando la probabilidad condicional de patrones de subidas y bajadas seguidas, en intervalos de 1 segundo a 60 minutos. Estos intervalos son habitualmente utilizados en el conocido como HFT (*High Frecuency Trading*). Los autores concluyen que existen ineficiencias en el mercado, para intervalos de tiempo menores a 30 minutos, estadísticamente significativas. Para períodos superiores a 30 minutos el comportamiento del mercado parece ser compatible con la HME.

En (Duarte and Perez-Iñigo, 2013) se hace una revisión de la eficacia de los mercados en distintas zonas geográficas, llegando a conclusiones similares: las mayores ineficacias de mercado se producen en mercados emergentes, siendo el mercado europeo y estadounidense más eficiente, principalmente a partir de 1990.

En (Manahov et al., 2014) se evalúa la capacidad del HFT en operaciones de 1 minuto, para generar retornos extraordinarios en el FX (mercado de Divisas). Los autores encuentran rendimientos superiores a los esperables por la HME, incluso aplicando costos de transación. Los resultados son tanto estadística como económicamente significativos. La práctica del HFT parece aportar eficiencia a los precios del mercado donde opera.

En (Yeh and Hsu, 2014) se estudia el Mercado de Valores de Taiwan, en los años 2005 a 2009. El artículo concluye que existen evidencias fiables de que dicho mercado sobre reacciona. En base a ello, los autores presentan un modelo de decaimiento exponencial que consideran más cercano a la realidad que la HME.

De los trabajos enumerados en este apartado, donde se cuestiona la validez de la HME, se puede concluir que si bien en líneas generales la HME parece ser cierta existen también muchas evidencias de ineficiencias en ámbitos temporales o geográficos concretos. Como razones aportadas para explicar la predictibilidad de las cotizaciones

se pueden citar: la sobrereacción de los inversores, la irracionalidad en el corto plazo de éstos, la difusión de información de forma gradual y el efecto manada.

Sin embargo, pocos estudios pueden demostrar, de forma estadísticamente robusta, la presencia constante y cuantificable de ineficiencias. Posibles razones a esta falta de contundencia pueden ser: 1. El dinamismo del mercado que deja obsoletas reglas o ineficacias muy deprisa y 2. La información aportada por dichas ineficacias quizás explique solo una pequeña parte de los movimientos, siendo la mayor parte aleatorios.

2.1.1. Aprendizaje Automático para predicción del mercado

A continuación se presentan trabajos, en orden cronológico, de técnicas de Aprendizaje Automático que se han utilizado en el contexto de la inversión en el mercado de valores.

Como datos de entrada a los clasificadores se han utilizado: 1. Datos econonómicos contables, 2. Indicadores técnicos (ver Sección 2.1.2) y 3. Descriptores sencillos de series históricas (ej. promedios, máximos y mínimos).

En una de las primeras publicaciones de apliciones del Perceptrón Multicapa (MLP) en este campo, Kimoto (Kimoto et al., 1990) aplica dicha técnica a la predicción del índice del Mercado de Valores de Tokyo.

En (Zhang et al., 1998) los autores hacen una revisión de aplicaciones de MLP a la inversión en mercados.

Por su parte, en (Mizuno et al., 1998) se afirma poder alzanzar un 63 % de precisión de las cotizaciones de acciones en el Mercado de Valores de Tokyo.

En (Huang et al., 2005) se concluye que la técnica Máquinas de Vectores Soporte (SVM) pueden resultar útil para predecir la dirección del cambio (sube o baja) del Nikkei Index (Índice del Mercado de Valores Japonés). No obstante, apuntan que realizar predicciones acertadas es muy costoso, dado que que el mercado parece complejo, evolutivo, dinámico no-lineal y está afectado por interacciones con noticias políticas, condiciones económicas y expectativas subjetivas de los diferentes actores del sistema.

En (Wong and Schulenburg, 2007; Schulenburg and Ross, 2001, 2000, 1999) se presentan diferentes técnicas (LCS, XCS) que, haciendo uso del concepto de adaptación continua del modelo predictivo al dinamismo del mercado, permiten obtener información útil de éste.

En (Chen and Navet, 2007) los autores intentan dilucidar si los fracasos publicados en la predicción del *timing* (elección del mejor momento de para realizar una operación en el mercado), haciendo uso de programación genética, se deben a errores o carencias de la técnica de predicción o a la aleatoriedad intrínseca del mercado.

En (Qian and Rasheed, 2007) se intenta predecir el *Dow Jones Industrial Average Index* haciendo uso de MLP, Árboles de Decisión y K-NN. El estudio concluye que no todos los períodos evolucionan de forma igualmente aleatoria. Los autores hacen uso del porcentaje de aciertos para medir la bondad de la predicción, alcanzando valores de hasta un 65 %.

En (Huang et al., 2008) se propone un algoritmo de selección de atributos, para problemas de predicción en Mercados de Valores Asiáticos. Como datos de entrada los autores utilizan 23 indicadores técnicos (ver Sección 2.1.2), calculados a partir de datos del año 1991, para los mercados de Korea y Taiwan. El objetivo es predecir la dirección del movimiento del índice (sube o baja). El algoritmo propuesto se basa en un sistema de voto, donde haciendo uso, como técnica de selección de atributos, de un $Wrapper$ (ver Ec. 2.3) y diferentes clasificadores (SVM, K-NN, MLP, Random Forest y Regresión Logística), se obtiene el subconjunto óptimo de atributos. Se entiende aquí subcojunto óptimo como aquel que permite obtener los mejores resultados en fase de clasificación. Como medida del éxito de la clasificación los autores hacen uso de *accuracy*. El artículo concluye que la solución propuesta (uso de una fase de selección de atributos previa a la clasificación) obtiene mejores resultados que utilizar como entrada del clasificador todos los indicadores.

En (Maragoudakis and Serpanos, 2010) se presenta el método Markov Blanket Random Forest para predecir el mercado de valores. Los autores afirman que supera a la estrategia pasiva *Comprar y Mantener* y a otros métodos de regresión utilizados en el experimento (Regresión Lineal, SVM y MLP).

En (Naeini et al., 2010) se presenta un sistema para predicción del mercado de valores basado en redes neuronales MLP y *Elman recurrent network* (ERN).

En (Alkhatib et al., 2013) se utiliza K-NN para predecir la evolución de cinco compañías del Mercado de Valores de Jordania durante el año 2009. Como datos de entrada al algoritmo se usan los *Ticks Data* (ver Sección 4.2.1) diarios de cada valor.

En (Qin et al., 2013) se realizan prediciones sobre el Mercado de Valores de Singapour. Como técnica de regresión se ha utilizado Gradient Boosted Random Forest

(GBR). El método presentado supera a la estrategia *Comprar y Mantener* en rendimiento económico.

En (Ładyżyński et al., 2013) se presenta un nuevo método, basado en Random Forest, que presenta la característica de la adaptación continua a series financieras no estacionarias.

En (Booth et al., 2014) se estudia el rendimiento de Random Forest y la estacionalidad sobre datos del Mercado de Valores Alemán (DAX). Los autores plantean un experimento donde Random Forest es entrenado cada 50 días y utilizado para predecir el valor exacto del precio de una acción. El experimento se plantea, por tanto, como un experimento de regresión donde el objetivo es predecir el precio exacto futuro de la acción. El experimento abarca 30 acciones, desde el 2000 al 2013. Como datos de entrada al algoritmo se utilizan tanto *Ticks Data* como 30 indicadores técnicos calculados a partir de éstos. La fase de clasificación es precedida por una de selección de atributos, la cual haciendo uso de Random Forest va eliminando atributos para optimizar el *Root Mean Squared Error* ($RMSE$), calculado haciendo uso de Validación Cruzada. Los autores concluyen que el modelo presentado mejora los resultados de la estrategia pasiva *Comprar y Mantener* tanto desde el punto de vista económico como en términos de porcentaje de aciertos.

En (Patel et al., 2015) se intenta predecir la dirección del movimiento (sube o baja) del Indian Stock Market Index. El experimento abarca desde el 2003 al 2012. Como datos de entrada preprocesan los *Ticks Data* del mercado para obtener hasta 10 indicadores técnicos. En el artículo se comparan los resultados de procesar los indicadores técnicos como valores continuos frente a discretizar éstos en solo dos valores (+1, -1). Como medida de los resultados se ha utilizado el porcentaje de aciertos. Como técnicas de clasificación se han testado Random Forest, SVM, Redes Neuronales y Naive-Bayes. Los autores concluyen que se obtienen mejores resultados discretizando los indicadores técnicos que utilizando los valores continuos. También concluyen que Random Forest permite obtener los mejores resultados en este experimento.

En (Ballings et al., 2015) los autores presentan una evaluación de diferentes técnicas de clasificación para intentar predecir la dirección del cambio de precios de acciones del Mercado de Bolsa Europeo. El estudio abarca 5.700 empresas a lo largo de todo el año 2014. En el artículo se compara el rendimiento de varios métodos de los llamados Ensambles (Random Forest, Ada y Kernel Factory) y las técnicas tradicionales (MLP,

Logistic Regresion, SVM y K-NN). Como datos de entrada a los algoritmos, se han creado 81 indicadores que contienen datos financieros y contables de cada empresa. Como medida del rendimiento se ha usado el área bajo la curva ($AUC = \int_0^1 \frac{TP}{P} d\frac{FP}{N}$).

El objetivo ha sido comparar si el uso de las técnicas de Aprendizaje Automático han dado lugar a rendimientos por encima de los esperables por azar. El estudio ha analizado los resultados globales para todo el período experimental y no se ha tenido en cuenta la evolución en el tiempo. Los autores concluyen que varias técnicas alcanzan rendimientos superiores al azar y, de éstas, Random Forest es la que obtiene los mejores resultados.

Como **conclusión final** a esta sección de estudio de la predictibilidad del mercado de valores, se puede plantear como hipótesis que en los datos de las cotizaciones históricas existe cierta cantidad de información utilizable para prededir el movimiento futuro del mercado. Si bien, para explotar adecuadamente dicha información, parece ser necesario superar algunas limitaciones de los enfoques aquí tratados.

Se considera oportuno considerar el mercado como un sistema dinámico donde las reglas o ineficiencias aparecen y desaparacen de forma contínua y rápida en el tiempo. Por ello, el sistema a desarrollar debería implementar algún sistema de aprendizaje continuo donde se extraiga información de la historia cercana, se la dé un uso breve y se deseche a favor de nuevas reglas de forma continua.

El segundo elemento a tener en cuenta es el siguiente: la información presente en los datos históricos parece ser de poca cuantía, por ello, parece ser realista aspirar a clasificar correctamente solo unos pocos patrones por encima de los esperables por azar y no pretender clasificar correctamente todos los patrones del problema. Ésto condiciona varios elementos de diseño de las nuevas técnicas, entre otros, la medida de la bondad del proceso. Por ello, parece razonable pensar que nuestro objetivo debe ser más realizar predicciones muy precisas, aunque abarquen pocos patrones, que realizar predicciones que abarquen un gran número de patrones, donde por la naturaleza de los datos es muy difícil que se consiga por parte de un modelo dado predecir la totalidad de los movimientos del mercado. Nuestro sistema no será capaz de predecir como se moverán todos y cada uno de los valores en el futuro pero, sí podría ser posible predecir con cierto grado de precisión qué harán unos pocos valores en ciertos momentos concretos.

Así, se puede establecer como objetivo de las nuevas técnicas, localizar o desarrollar reglas que expliquen durante cierto tiempo y de la forma más precisa la ocurrencia

futura de algún evento de interés (por ejemplo, subida de una cotización) con un nivel de fiabilidad o confiabilidad arbitrariamente alto.

2.1.2. Indicadores Técnicos

En el ámbito de *trading*, podemos definir Indicador Técnico como una fórmula matemática que, aplicada a unos datos históricos de mercado, es capaz de calcular un índice que indique algún comportamiento futuro útil: próxima subida, sobrecompra de un valor, etc. El desarrollo del Análisis Técnico actual debe sus inicios a los principios enumerados en la Teoría de Dow. Las premisas principales, enumeradas por Hamilton, Rhea y Schaefer y actualizadas por (Brown et al., 1998) son: 1. La cotización es un reflejo absoluto de todas las noticias y las fuerzas de los mercados, 2. Las cotizaciones se mueven según tendencias, 3. Las tendencias son confirmadas con volumen, 4. El mercado se mueve en tres movimientos: fase primaria, secundaria y menor, 5. Las tendencia principal de un mercado alcista tiene tres fases: fase de acumulación, fase de participación pública y exceso de fase. En el caso de un mercado bajista las fases serían: fase de distribución, fase de participación pública y fase de pánico y, 6. Las tendencias existen hasta que exiten señales que prueban que han terminado.

En esta sección se van a presentar, organizados en familias, algunos de los indicadores técnicos más utilizados y se van a apuntar las publicaciones que apoyan su posible eficiencia.

- Velas Japonesas (Fischer and Fischer, 2003; Nison, 2004; Bigalow, 2011): Hikkake, Morning start, Doji, Hammer. La técnica se desarrolló originariamente hace varios siglos, en Japón, para predecir las fluctuaciones del precio del arroz.
- Soporte y Resistencia (Brock et al., 1992; Kavajecz and Odders-White, 2004; Osler, 2000; Fernández Rodríguez et al., 1999; Wyckoff, 1910), rotura de canales: Botton, Ondas de Elliot (Bhattacharya and Kumar, 2006), Fibonacci retracement (Fischer and Fischer, 2003; Batchelor and Ramyar, 2006), Pivot Point. Estas técnicas intentan predecir suelos y techos de la cotización.
- Tendencia (Wilcox and Crittenden, 2005; Chaudhuri and Wu, 2003; Alexander, 1961; Covel, 2004; Fung and Hsieh, 2001; Rangvid, 2001): ADX, CCI, MAC, Media Móvil (Fang and Xu, 2003; Gartley, 1935). Intentan sacar partido a los efectos de la autocorrelación de los precios.

- Momento (Connolly and Stivers, 2003; Okunev and White, 2003; Chan et al., 2000; Grundy and Martin, 2001): MFI (Flujo del dinero), RSI (Fuerza relativa), Oscilador Escotastico , TSI (Verdadero índice de fuerza), Williams. Intentan estimar la fuerza o persistencia de un movimiento o tendencia.

- Volumen (Chen et al., 2001; Huddart et al., 2005; Lee and Swaminathan, 2000): Acumulación/Distribución, EMV, FI, NVI, OBV Granville (1963), PCR, VPT (Volumen/tendencia). Realizan predicciones basándose en el volumen de transaciones.

- Volatilidad (Balsara et al., 2007): ATR (verdadero promedio de rango), Bandas de Bollinger[1], Donchian channel, VIX (volatilidad de mercado). Intentan detectar cambios bruscos en las cotizaciones.

- Breadth (Chen et al., 2002; Slaughter, 2002): ADL (línea de avance retroceso), TRIN, oscilador McClellan. Estiman la sobrecompra o sobreventa de activos, atendiendo al dinero que entra o sale del mercado.

2.2. Aprendizaje supervisado

Dado un conjunto de N ejemplos de entrenamiento de la forma $\{(x_1, y_1), ..., (x_N, y_N)\}$, siendo X_i el vector de entrada del ejemplo i e Y_i la etiqueta o clase, el objetivo del aprendizaje es encontrar una función $g_w : X \to Y$, denominada modelo, que represente la correspondencia existente entre los vectores de entrada y el valor de salida correspondiente. Para obtener g_w se puede utilizar una función de *scoring* $f : X \times Y \to R$ de forma que definamos $w = \arg\max_w f(g_w(X), Y)$. Como función f podremos usar una función de utilidad a maximizar o, una función de error o de coste a minimizar. En esta tesis, consideraremos funciones de estilo utilidad y, por tanto, el criterio será siempre maximizar. Con ello, se puede asimilar el aprendizaje a un proceso que tenga como objetivo encontrar el modelo, valores de w, que hagan máxima la función de *scoring* f.

Para encontrar el valor óptimo de w, algunas técnicas de aprendizaje automático utilizan un procedimiento formal de optimización llamado Descenso por Gradiente Estocástico (SGD) (Friedman, 2002, 1987). Éste permite encontrar los mínimos/máximos

[1]Los estudios revisados encuentran evidencias de utilidad del indicador de Bandas de Bollinger, si bien, para valores contrarios a los proporcionados por el indicador.

valores de una función siempre y cuando esta sea una función derivable. Por ejemplo, la técnica del Perceptrón Multicapa usa dicho algoritmo para entrenamiento de los pesos de las neuronas.

Otras técnicas utilizan una función generalista (por ejemplo: $IGAIN$, *Entropia* o $GINI$). Ejemplos de estas técnicas serían los Árboles de Decisión C4.5 (Quinlan and Rivest, 1989; Quinlan, 1986; Quinlan et al., 1979) y Random Forest (Breiman, 2001).

En el caso de utilizar estas técnicas en un problema donde deseemos optimizar una función de *scoring* distinta (por ejemplo, quisiéramos optimizar la *precision* en la clase C_1) deberemos utilizar un procedimiento, externo a la técnica, donde variando el valor de algunos de los parámetros del clasificador, podamos seleccionar la combinación de éstos que optimize la medida adecuada al problema a resolver.

Para encontrar el valor adecuado de cada parámetro podemos evaluar el resultado del clasificador para diferentes valores entre un rango dado. Ejemplos de este sistema podemos encontrarlos en los metaclasificadores de WEKA (Hall et al., 2009): MultiSearch, GridSearch y CVParameterSelection.

Para guiarnos en la búsqueda de los mejores parámetros podemos, también, utilizar procedimientos basados en el espacio ROC (Hanley and McNeil, 1982) o, alternativamente, en la curva *precision/recall* Figura 2.1.

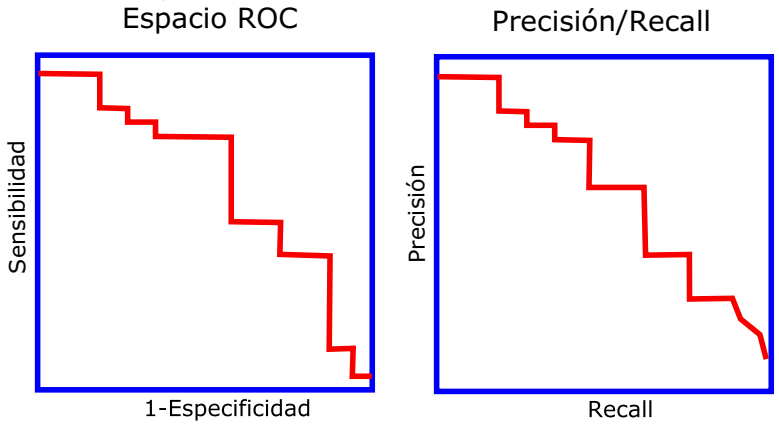

Figura 2.1: Espacio ROC y Curva Precisión/Recall

.

Ambas curvas pueden obtenerse a partir de la matriz de contingencia de una clasificación. En problemas binarios, si llamamos TP(Verdadero Positivo), FP(Falso Positivo), TN(Verdadero Negativo) y FN(Falso Negativo), podemos definir las siguientes

funciones:

$$Sensibilidad = \frac{TP}{TP + FN} \tag{2.1}$$

$$Especificidad = \frac{TN}{TN + FP} \tag{2.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Dos ejemplos de técnicas que usan la curva ROC para optimizar el resultado de clasificadores y ensambles de clasificadores se presentan en (Gao et al., 2006) y (Pietraszek, 2007).

Existen factores que dificultan el éxito del aprendizaje automático, como pueden ser el desbalanceo de clases y la alta dimensionalidad en el espacio de entrada.

El **desbalanceo de clases** consiste en que el número de patrones de cada clase en el *conjunto de ejemplos* no está equilibrado, por lo que unas clases están sobrerrepresentadas respecto a otras.

De la revisión bibliográfica realizada en relación al desbalanceo de clases en problemas de clasificación se deduce que, básicamente, todas las soluciones publicadas sobre esta cuestión, siguen alguna de estas aproximaciones (Eitrich et al., 2007):

- *Oversampling*: añade al conjunto de entrenamiento tantos patrones repetidos de la clase minoritaria como sea necesario para equilibrar el número de patrones entre clases. Esta aproximación tiene como desventajas la sobrecarga el procesamiento con patrones repetidos y la pérdida de información relativa a probabilidades y densidades (Barandela et al., 2004).

- *Undersampling*: desecha patrones de la clase mayoritaria hasta que el total de patrones de cada clase se equilibra. la principal desventaja es la pérdida de información (Akbani et al., 2004).

- *Threshold*: recalcula los umbrales usados por la técnica de clasificación de forma que la frontera entre clases se acerque más o menos a sus agrupaciones dependiendo de si es la clase minoritaria o no.

- Algoritmos basados en calcular penalizaciones o utilidades diferentes para cada clase (Zhou and Liu, 2006). Como ejemplo de técnica en esta categoría se puede citar Metacost (Domingos, 1999), basado en matriz de costes.

No parece existir una aproximación claramente mejor que las demás. Todas ellas presentan ventajas y desventajas. En cualquier caso, podemos pensar que toda técnica de Aprendizaje Automático que pretenda ser de utilidad con conjuntos de datos desbalanceados debe poder gestionar este problema. Ya sea balanceando previamente los datos (*Oversampling*, *Undersampling*) o gestionando el desbalanceo en el propio sistema de valoración de soluciones de la técnica (Threshold, Metacost o similar).

Otro de los factores que dificulta el aprendizaje es la **alta dimensionalidad** en el espacio de entrada. Este concepto hace alusión a la presencia de un número muy alto de atributos en el problema a tratar. Cuando se trabaja con espacios de cientos o miles de dimensiones se observa un efecto llamado *Maldición de la Dimensionalidad* (Shalev-Shwartz and Ben-David, 2014; Hastie et al., 2003; Friedman et al., 2001; Hughes, 1995, 1968; Bellman, 1961). Se manifiesta de muchas maneras pero, en resumen, viene a decir que el número de patrones de entrenamiento necesarios para generalizar correctamente crece exponencialmente con el número de dimensiones. Este factor afecta muy negativamente a cualquier tratamiento estadístico y de clasificación que sobre los datos quiera hacerse (Jimenez and Landgrebe, 1998).

Cuando el número de atributos es muy alto en relación al número de muestras o patrones de entrenamiento, se puede producir el efecto conocido como sobreajuste. Éste afecta muy negativamente al rendimiento de la mayor parte de las técnicas automáticas ya que, provoca que modelos que presenten errores de entrenamiento bajos dan lugar a errores de test altos. Así, el sobreajuste provoca que la técnica en cuestión generalice mal.

Existen trabajos empíricos y analíticos (Hwang et al., 1994; Scott, 1992; Fukunaga and Hayes, 1989; Jain and Waller, 1978) que caracterizan el número óptimo de atributos a tratar frente al número de patrones de entrenamiento.

En la mayor parte de los casos, los patrones disponibles vienen limitados por el *conjunto de entrenamiento* y no es posible generar el número de patrones necesario para abordar directamente el problema.

Antes de tratar este tipo de problemas conviene aplicar técnicas de selección de atributos. La siguiente sección presenta una revisión de los principales algoritmos. Éstos

tienen como objetivo seleccionar un subconjunto de atributos, tal que se mantenga la mayor parte de la información, descartando los atributos redundantes o irrelevantes.

2.3. Selección de atributos

Dentro del Aprendizaje Automático, el área de la Selección de Atributos se especializa en la tarea de seleccionar el subconjunto de atributos que pueden resultar de máxima utilidad para las tareas de clasificación o regresión posteriores.

Existe una gran cantidad de trabajos publicados sobre técnicas de selección de atributos. Revisiones de muchas de las aproximaciones clásicas, se pueden consultar en: (Liu et al., 2009; Saeys et al., 2007; Guyon and Elisseeff, 2003; Forman, 2003). Por su parte, (Liu and Yu, 2005) presenta una revisión detallada de algoritmos de selección de atributos específicos para problemas de clasificación.

La selección de atributos puede aportar varios beneficios: reducción de $overfitting$, mejora de la precisión de las predicciones, eliminación de atributos redundantes, simplificación de los modelos y reducción del tiempo de proceso (Liu et al., 2014).

El objetivo que persiguen las técnicas de selección de atributos es obtener la lista de atributos más relevantes de un conjunto de datos dado. Si dicho proceso tiene éxito, un subconjunto de atributos nos aportará la misma información que el conjunto original, deshaciendonos, por el camino, de los atributos irrelevantes o redundantes que pudieran existir en los datos originales.

En la bibliografía no existe una única definición formal de relevancia. En este trabajo, se va a seguir la línea iniciada por (Blum and Langley, 1997). Según este autor, las definiciones de relevancia deben tener en cuenta la pregunta: ¿Relevante para qué?.

Definiciones de relevancia:

- Relevancia con respecto a la clase: un atributo a_i es relevante para la clase Y, si existe, al menos, un par de ejemplos, e_j y e_k, que difieran solo en el valor de a_i y Y_k es distinto a Y_j.

- Relevancia entrópica: se define como la información mutua entre el atributo y la clase.

- Relevancia como utilidad incremental: dado un conjunto de atributos A, un algoritmo de aprendizaje L, un subconjunto de atributos S ($S \subset A$), un atributo a_i

($a_i \in A$) es incrementalmente útil, si la tasa de acierto que la hipótesis que L produce usando el subconjunto de atributos $\{a_i\} \cup S$, es mejor que la tasa obtenida utilizando solo el subconjunto S.

En cualquier caso, dado que la bondad del subconjunto de atributos seleccionados se mide según un criterio específico, comúnmente dependiente de la aplicación que se quiera dar al mismo, no siempre será posible obtener un subconjunto de atributos que optimice cualquier criterio dado. Así, los atributos que compongan el subconjuto óptimo según un criterio no necesariamente serán los óptimos para optimizar otro criterio distinto (Liu and Yu, 2005).

Los métodos de selección de atributos pueden agruparse, atendiendo al tipo de evaluación, en tres categorías (Guyon and Elisseeff, 2003; Das, 2001): $Filters$, $Wrappers$ e Híbridos:

- *Wrappers* (Liu and Yu, 2005): combinan la búsqueda en el espacio de atributos con el algoritmo de aprendizaje, evaluando subconjuntos de atributos en base al comportamiento en la fase de clasificación. Así, realizan la selección evaluando los resultados que el uso de éstos por un clasificador dado permite obtener. Dado que el proceso de búsqueda es guiado por el rendimiento en clasificación, suelen obtener buenos resultados (Somol et al., 2005; Dy and Brodley, 2004; Kohavi and John, 1997). Como desventajas, los métodos $Wrappers$ pueden padecer el problema del $overfitting$ (Loughrey and Cunningham, 2005; Reunanen, 2003), especialmente si el conjunto de entrenamiento es pequeño y la búsqueda sea exhaustiva. Además, el uso de $Wrappers$ suele implicar costosos cálculos (Inza et al., 2004). Algunas técnicas de selección de atributos representativas de este enfoque son: RC (Domingos, 1997), ELSA (Kim et al., 2000), GA (Vafaie and Imam, 1994) y RVE (Stracuzzi and Utgoff, 2004).

- *Filters*: el procedimiento de selección es realizado utilizando características generales de los datos para evaluar los atributos independientemente del proceso de clasificación (Liu and Yu, 2005). Suelen ser muy rápidos y, en algunos contextos, razonablemente precisos (Kohavi and John, 1998). Técnicas representativas de esta categoría son: CFS-Subset (Hall, 1999; Witten et al., 2011), ReliefF (Kira and Rendell, 1992b) y Chi-Squared(Liu and Setiono, 1995).

- Híbridos: utilizan diferentes criterios de evaluación en las diferentes etapas de búsqueda. Aúnan lo mejor de los modelos anteriores (Das, 2001; Xing et al., 2001). A menudo, utilizan una primera fase muy rápida, utilizando una aproximación *Filter*, seguida de una segunda donde se afina la lista de atributos en base al resultado de la clasificación. Como ejemplos de técnicas que utilizan esta aproximación se puede citar a: BBHFS (Das, 2001) y Xing's (Xing et al., 2001).

Atendiendo a cuántos atributos evalúan simultáneamente las técnicas de selección de atributos pueden ser clasificadas también en: *Weighting algorithms* y *Subset search algorithms*. Los primeros evalúan atributos individuales y los segundos subconjuntos de atributos (Uysal, 2016; Yu and Liu, 2004, 2003).

Una desventaja que se puede apuntar sobre las técnicas que evalúan los atributos de uno en uno consiste en que éstas no son capaces de eliminar los atributos redundantes ni de valorar las interacciones entre atributos (Karegowda et al., 2010).

Un método de selección de atributos de evaluación de subconjuntos (*Subset search algorithms*) requiere, típicamente, de los siguientes elementos (Aguilar-Ruiz and Diaz-Diaz, 2005; Lorenzo Navarro, 2001; Dash and Liu, 1997): un procedimiento de búsqueda para generar subconjuntos de atributos candidatos, un método de evaluación de subconjuntos para poder comparar entre éstos y, un criterio de parada de la búsqueda.

Los procedimientos habituales de **búsqueda** para generación de subconjuntos de atributos son:

- Búsqueda completa o exhaustiva: garantiza encontrar el óptimo para la función de evaluación elegida (Liu and Yu, 2005; Cover and Van Campenhout, 1977). Desafortunadamente, la búsqueda exhaustiva es un problema NP-Difícil (Liu and Yu, 2005; Kohavi and John, 1997; Davies and Russell, 1994). Como ejemplos de técnicas que utilizan esta búsqueda se podrían citar: B & B (Nakariyakul and Casasent, 2007; Chen, 2003; Narendra and Fukunaga, 1977), BS (Doak, 1992), AMB & B (Foroutan and Sklansky, 1987), FSLC (Ichino and Sklansky, 1984a) y FSBC (Ichino and Sklansky, 1984b).

- Búsqueda secuencial (Liu and Motoda, 2012): utilizan una aproximación *greedy* para ampliar o reducir subconjuntos de atributos con el fin de maximizar un criterio dado. Pueden funcionar hacia adelante, añadiendo atributos, hacia atrás,

eliminándolos o, añadiendo y eliminando atributos de forma alternativa. Como ejemplos de esta aproximación pueden citarse: DEFS (Khushaba et al., 2011), FS (Pudil et al., 1994), ReliefF (Kira and Rendell, 1992b) y a CFS-Subset (Hall, 1999; Witten et al., 2011).

- Búsqueda aleatoria: utilizando diferentes grados de aleatoriedad, intentan superar mínimos locales que podrían limitar las posibilidades de una búsqueda puramente secuencial para encontrar el subconjunto óptimo de atributos. Como posibles aproximaciones a este tipo de búsqueda podrían citarse los algoritmos Templado simulado (Doak, 1992) y Las Vegas (Brassard and Bratley, 1996). Como técnicas de selección de atributos que utilizan este método de búsqueda, se puede citar: LVI (Liu and Motoda, 2012), QBB (Liu and Motoda, 2012), LVF (Liu et al., 1996) y RVE (Stracuzzi and Utgoff, 2004).

Los métodos *Wrapper* utilizan el resultado de un clasificador para evaluar cada subconjunto de atributos. Por contra, los métodos *Filter* necesitan de una medida independiente para evaluar éstos. Las **medidas de evaluación** más utilizados, por los *Filters*, para comparar subconjuntos de atributos son:

- Medidas de distancia (Liu et al., 2002): también conocidas como separabilidad, divergencia o discriminación estas medidas intentan valorar la diferencia en las probabilidades condicionales respecto a la clase, obtenidas al usar cierto subconjunto de atributos. Las técnicas FSDD (Liang et al., 2008) y ReliefF (Kira and Rendell, 1992b) utilizan este criterio de evaluación.

- Medidas de información (Yao et al., 1999; Cardie, 1993): intentan valorar la ganancia de información que puede proporcionar un subconjunto de atributos dados. La ganancia de información se puede medir haciendo uso del concepto de Entropía (Ben-Bassat, 1982), de la Entropía Condicional o de la Información Mútua (Bennasar et al., 2015). Ejemplo de técnica que utiliza estas medidas: Chi-Squared (Liu and Setiono, 1995).

- Medidas de dependencia (Mucciardi and Gose, 1971): son también conocidas como medidas de correlación o similitud. Miden la capacidad predictiva del valor de la clase a partir del valor de cada atributo o subconjunto de éstos. La técnicas FCBF

(Yu and Liu, 2003) y CFS-Subset (Hall, 1999; Witten et al., 2011) utilizan este criterio.

- Medidas de consistencia (Ruiz et al., 2002): asignan el valor máximo de evaluación al subconjunto mínimo de atributos que es capaz de separar los patrones, según la clase, al menos tan bien como el conjunto total de atributos. Como ejemplos de técnicas que usan este criterio puede nombrarse: FOCUS (Almuallim and Dietterich, 1991), LVI (Liu and Motoda, 2012), QBB (Liu and Motoda, 2012) y LVF (Liu et al., 1996).

Finalmente, es necesario definir un **criterio de detención o parada** para el algoritmo de búsqueda. Los criterios más utilizados son:

- Búsqueda completa terminada: en el caso de búsquedas exhaustivas el algoritmo de búsqueda finaliza cuando ha evaluado todas las posibidades.

- Umbral de la función de evaluación: el algoritmo finaliza cuando se alcanza un valor de evaluación de alguna de las soluciones superiores a un umbral dado.

- Fin de la búsqueda cuando no existe mejora al añadir o eliminar atributos: en búsquedas secuenciales llegar a un punto en el que ni la adicción de atributos ni la eliminación de éstos permite una mejora en el valor de la medida de evaluación.

Selección de atributos en problemas de alta dimensionalidad: revisiones de técnicas de selección de atributos utilizadas en problemas de Alta Dimensionalidad pueden ser consultadas en: (Ferreira and Figueiredo, 2012; Hastie et al., 2009; Ruiz et al., 2009; Guyon et al., 2008; Guyon and Elisseeff, 2003).

Dado que los algoritmos de tipo $Wrapper$ utilizan un clasificador para evaluar subconjuntos de atributos suelen implicar altos costos computacionales (Inza et al., 2004), lo que las convierte en poco adecuadas para este tipo de problemas. Además, el número de patrones en este tipo de problemas resulta, a menudo, insuficiente (Hua et al., 2009) para soslayar el problema del $overfitting$ de los $Wrapper$ y la Maldición de la Dimensionalidad. Por ambos motivos, en problemas con un gran número de atributos es difícil recomendar técnicas de selección de atributos de tipo $Wrapper$ (especialmente las que combinan este sistema de evaluación con búsquedas de tipo exhaustivo). En

general, soluciones de tipo *Filter* o de tipo Híbrido parecen ser más adecuadas para este tipo de problemas (Ferreira and Figueiredo, 2012).

No obstante, existen algunas soluciones que, aún siendo *Wrapper*, implementan mecanismos diseñados para reducir el número de evaluaciones totales necesarias, por lo que pueden ser aplicables a este tipo de problema: FSDD (Liang et al., 2008), GRASP (Nakariyakul and Casasent, 2007) y DEFS (Khushaba et al., 2011).

Como soluciones Híbridas, ya sea combinando una primera fase *Filter* con una segunda *Wrapper* o siguiendo aplicando un diseño embebido, podemos citar a: (Bermejo et al., 2011), BLogReg (Cawley and Talbot, 2006), SMLR (Krishnapuram et al., 2005), JCFO (Krishnapuram et al., 2005), SLogReg (Shevade and Keerthi, 2003) y (Xing et al., 2001).

Por último, como implementaciones, de tipo *Filter*, recomendadas en problemas de alta dimensionalidad (Ferreira and Figueiredo, 2012) se pueden citar: FCBF (Yu and Liu, 2003), CFS-Subset (Hall, 1999), ReliefF (Kira and Rendell, 1992b), mrMR (Peng et al., 2005), CoFS (Sun et al., 2012) y, en general, la familia de métodos de selección de atributos basados en la teoría de la información (Brown et al., 2012).

(Bolón-Canedo et al., 2013) presenta una interesante revisión de métodos de selección de atributos y una **metodología para compararlos** utilizando conjuntos de datos sintéticos. El artículo estudia el desempeño de diferentes métodos de selección de atributos, en presencia de cantidades crecientes de atributos irrelevantes, ruido en los datos, redundancia y ratio reducido entre número de patrones y número de atributos. Los autores resaltan la ventaja de usar conjuntos de datos sintéticos, para estudiar el comportamiento de cada método frente a cada factor de dificultad, por la posibilidad de conocer *a-priori* el conjunto óptimo de atributos a seleccionar.

Para efectuar la comparación entre los métodos de selección el artículo estudia el tiempo de proceso que necesita cada método para completar las pruebas, una medida sintética llamada *Index Success* (ver Ec. 2.5) y el porcentaje de aciertos o *Accuracy* (ver Ec. 2.6), obtenida por diferentes clasificadores haciendo uso de los atributos seleccionados por cada técnica. Es de destacar, que los autores han concluido que ReliefF ha sido el método más resistente, de entre los probados, al factor ruido. Por su parte, CFS-Subset ha resultado ser el más sensible a este factor.

En la Ecuación 2.5 se muestra la definición de $IndexSuccess$. R_T representa el total de atributos relevantes de un problema. R_S es el número de atributos seleccionados.

I_S es el número de atributos irrelevantes seleccionados. Por último, I_T es el número de atributos irrelevantes totales.

$$IndexSucces = \frac{R_S}{R_T} - \alpha \times \frac{I_S}{I_T} \times 100 \ , \ \alpha = min(\frac{1}{2}, \frac{R_T}{I_T}) \tag{2.5}$$

En la Ecuación 2.6 se define *Accuracy*. TP es el número de verdaderos positivos. TN representa el número de verdaderos negativos. Y, P y N, el número total de patrones positivos y negativos, respectivamente.

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.6}$$

Conclusiones para el diseño de los nuevos métodos:

A partir del estudio sobre algoritmos de selección de atributos realizado en esta Sección 2.3 se observa que existe una gran variedad de trabajos en la bibliografía para abordar esta tarea.

Dado que una de las características del problema a resolver en esta tesis es utilizar un número relativamente grande de atributos (entre algunos cientos y unos pocos miles), se considera que una aproximación *Filter* podría ser la más adecuada por su eficiencia en coste computacional.

Sería, además, muy conveniente que los nuevos métodos tengan en cuenta las posibles interaciones entre los diferentes atributos, por lo que un diseño de tipo *evaluación de subconjuntos* se considera como el más adecuado.

Dado que el problema a resolver presenta un moderado desbalanceo de clases y características probabilísticas se considera que las funciones de evaluación basadas en la teoría de la información (ej. Entropía) pueden resultar un buen punto de partida para el diseño de las nuevas técnicas.

Por último, para estudiar la robustez de los nuevos métodos frente a diversos factores de dificultad que puedan presentarse en los datos, un enfoque similar al utilizado en (Bolón-Canedo et al., 2013) parece ser adecuado. La combinación de experimentos sobre conjuntos de datos sintéticos (haciendo uso de medidas de éxito sintéticas) con otros sobre datos reales (utilizando medidas habituales para evaluar la bondad del resultado de clasificación, por ejemplo *G-means*) puede proporcionar una imagen muy precisa sobre las capacidades de los nuevos métodos.

2.4. Algoritmos utilizados en esta tesis

En esta sección se presentan los algoritmos de selección de atributos, de clasificación y de regresión utilizados en esta tesis para comparar con las técnicas propuestas.

Salvo donde se indique expresamente, como implementación de las técnicas aquí presentadas, se ha hecho uso de los algoritmos incluidos en el paquete WEKA (Hall et al., 2009).

2.4.1. Algoritmos de Selección de Atributos

A continuación se enumeran las técnicas de selección de atributos que se han utilizado en los experimentos.

- **CFS-Subset** (Hall and Smith, 1997; Hall, 1999; Witten et al., 2011): evalúa subconjuntos de atributos, encuandrándose así en la familia de técnicas *Subset Search Algorithms*. Hace uso de una medida de dependencia, considerando la habilidad predictiva individual de cada variable, así como el grado de redundancia entre ellas. Se utiliza asociado a un algoritmo buscador. Algunos métodos de búsqueda incorporados en WEKA son:
  - ExhaustiveSearch: realiza una búsqueda exhaustiva o completa por todo el espacio de atributos. Garantiza encontrar el subconjunto óptimo. El orden del algoritmo viene dado por $O(nAtrs^2 \times N)$, donde N es el número de instancias y $nAtrs$ el número de atributos de cada instancia.
  - BestFirst: pertenece a la familia de buscadores secuenciales *greedy*. Realiza una búsqueda hacia adelante o hacia atrás, a partir de un conjunto inicial, al que se van añadiendo o quitando atributos. Cada vez que la función de evaluación de subconjuntos empeora realiza un *backtraking*.
- **ReliefF** (Kira and Rendell, 1992b; Kononenko, 1994; Kira and Rendell, 1992a; Kononenko et al., 1997): Evalúa los atributos en base una medida de distancia, estudiando cómo sus posibles valores discriminan entre las diferentes clases y la cercanía a las otras clases. El orden del algoritmo viene dado por $O(nAtrs \times N^2)$, donde N es el número de instancias y $nAtrs$ el número de atributos de cada instancia.

- **Chi-Squared** (Liu and Setiono, 1995): pertenece a la familia de los *Weighting algorithms*, evalúa individualmente cada atributo, haciendo uso de una medida de información, calculando el valor del estadístico Chi-squared con respecto a la clase. El orden del algoritmo viene dado por $O(nAtrs \times \lg(N) \times N) \ldots O(nAtrs \times N^2)$, donde N es el número de instancias y $nAtrs$ el número de atributos de cada instancia.

El motivo de haber elegido estas técnicas frente a las que comparar las desarrolladas en esta tesis, ha sido contar con un abanico de técnicas que utilizan distintas medidas de evaluación y distintos procedimientos de búsqueda de subconjuntos.

Todas las técnicas elegidas son de tipo *Filter* ya que, a la vista de la revisión bibliográfica, parece que dicha aproximación es más adecuada para problemas con alto número de atributos.

CFS-Subset evalúa subconjuntos de atributos, utilizando una medida de dependencia. Con ello, además de valorar la habilidad predictiva individual de cada atributo, tiene en cuenta el grado de redundancia entre ellos. Para problemas con pocos atributos se ha usado *ExhaustiveSearch* como método de búsqueda de subconjuntos y, para problemas de más de 25 atributos, se ha usado *BestFirst*. Esta distinción se ha realizado porque el tiempo de proceso, utilizando *ExhaustiveSearch*, se dispara para problemas con más de unas pocas decenas de atributos.

ReliefF se ha elegido porque es un método estándar en la disciplina, generalmente con resultados bien posicionados y robusto a las interacciones entre atributos. Este método utiliza una medida de distancia como sistema de evaluación.

Por último, se ha hecho uso de **Chi-Squared**, que utiliza una medida de información para evaluar atributos individuales. Además de contar así con un método de los llamados *Weighting Algorithms*, Chi-Squared por estar basado en la teoría de la información, presenta paralelismos con las técnicas propias que hacen aconsejable su comparación.

2.4.2. Algoritmos de Clasificación

Las técnicas de clasificación que se han utilizado en los experimentos han sido:

- **Árboles de decisión (C4.5):**

Dado que existen algunos paralelismos entre este algoritmo y los métodos presentados en esta tesis, se van a describir en mayor detalle las ecuaciones en las que se basa por el interés que pueda tener para establecer comparaciones con las funciones de evaluación usadas en los nuevos métodos. Tanto los árboles de decisión como casi todas las funciones de evaluación propias hacen uso de conceptos provinientes de la Teoría de la Información. Además, uno de los métodos nuevos, LIA3, se basa en una estructura en árbol de decisión, por lo que se considera interesante detallar aquí los fundamentos de éstos.

(Quinlan, 1986; Quinlan and Rivest, 1989; Quinlan et al., 1979) presentó el método Induction Decision Trees (ID3). El método se basa en generar un árbol de decisión que, de forma recursiva y aplicando un criterio de división basado en la Ganancia de Entropía (ver Ec. 2.8), va subclasificando patrones hasta que llega a una hoja del árbol donde todos los patrones son de la misma clase.

Sea ε un conjunto de datos etiquetados con clases del conjunto Dom(Y)=$\{y_1,...,y_k\}$ y $frec(y_l,\varepsilon\ )$ el número de ejemplos de ε con clase y_l. Se define *Entropia*, grado de desorden del sistema, como:

$$Entropia(\varepsilon) = -\sum_{l=1}^{k} \frac{frec(y_l,\varepsilon)}{|\varepsilon|} \times \log_2 \frac{(frec(y_l,\varepsilon)}{|\varepsilon|}) \tag{2.7}$$

$$Ganancia(\varepsilon, X_i) = Entropia(\varepsilon) - \sum_{v=1}^{|X_i|} \frac{|\varepsilon(X_{iv})|}{|\varepsilon|} \times Entropia(\varepsilon(X_{iv})) \tag{2.8}$$

Como desventajas del método: muy a menudo, el árbol generado es muy grande y profundo. Además tales árboles pueden tener un bajo riesgo empírico (error en clasificación), pero su riesgo real (error en test) tiende a ser demasiado alto (Shalev-Shwartz and Ben-David, 2014). Dos soluciones pueden aplicarse para aliviar el problema: 1. Puede limitarse paramétricamente la profundidad máxima del árbol y 2. Una vez generado éste, puede realizarse un proceso de poda que intente mantener el error empírico pero haciendo uso de un árbol mucho más conciso. El propio Quinlan, desarrolló el método C4.5 (Quinlan, 1993), que es una evolución de ID3 y soluciona alguna de las limitaciones que tenía éste. El nuevo algoritmo, sustituye subárboles completos por hojas etiquetadas con la clase mayoritaria.

Desde su introducción ha sido un algoritmo muy exitoso debido a su robustez en los resultados y a su bajo coste computacional.

C4.5 utiliza la *razón de ganancia* como criterio de separación. De esta forma, soslaya el sesgo de ID3 a seleccionar atributos con muchos posibles valores. El algoritmo funciona en dos fases: una primera donde se crea el árbol haciendo uso de la *RazonDeGanancia* como criterio de partición y una segunda donde se efectua una poda del árbol para mejorar la generalización. El proceso completo puede entenderse como un algoritmo de optimización local, usado en la creación de cada nodo del árbol, seguido de un proceso de optimización global, aplicado en el proceso de poda.

$$InformacionDeSeparacion(\varepsilon, X_i) = -\sum_{v=1}^{|X_i|} \frac{|\varepsilon(X_{iv})|}{|\varepsilon|} \times \log_2 \frac{(\varepsilon(X_{iv})}{|\varepsilon|}) \tag{2.9}$$

$$RazonDeGanancia(\varepsilon, X_i) = \frac{ganancia(\varepsilon, X_i)}{InformacionDeSeparacion(\varepsilon, X_i)} \tag{2.10}$$

La representación del conocimiento en forma de árbol que conseguimos con estas técnicas, es muy intuitiva y cercana a la forma de organizar decisiones que tenemos los humanos. No obstante, la principal pega del uso de árboles es la dificultad computacional, en fase de entrenamiento, que puede calificarse como problema NP-Completo(Hyafil and Rivest, 1976).

En la parte experimental de esta tesis se ha hecho uso del algoritmo J48, incluido en WEKA, y que implementa la técnica C4.5.

- **Multi Layer Perceptrón (MLP)**

  (Rosenblatt, 1958) introdujo la idea del Perceptrón, que estaba inspirado en la forma en la que opera una neurona real. El Perceptrón es capaz de, a partir de un conjunto de patrones etiquetados, determinar la ecuación del plano discriminante que separa las diferentes clases del problema. Es así, una aproximación fundamentalmente geométrica al problema de la clasificación. (Rumelhart et al., 1985, 1988) presentaron el algoritmo Regla Delta Generalizada, (más conocida como Backpropagation), que permite apilar Perceptrones en diferentes capas (capa de

entrada, n capas ocultas, y capa de salida) y establece como propagar los errores por todas las capas de forma que todas ellas participen en el aprendizaje.

El Multi Layer Perceptrón (MLP) se considera un aproximador universal y puede gestionar relaciones no lineales entre los datos de entrada y salida. El objetivo final del aprendizaje consiste en minimizar el error cuadrático medio cometido por la red al clasificar un conjunto de datos.

Si elegimos una función de salida de la última capa que sea continua, podemos obtener además el grado de pertenencia de cada patrón a cada clase del problema.

Como limitaciones o desventajas de este método podemos citar: 1. Las funciones de transferencia de los elementos de procesado han de ser derivables y 2. La existencia de mínimos locales en la función de error dificulta converger al mínimo global de error.

- **K-Nearest Neighbors (K-NN)**

  Este clasificador (Cover and Hart, 1967) se basa en almacenar el *conjunto de entrenamiento* y cada vez que se requiere clasificar una nueva instancia p, se asigna la clase más frecuente en los K patrones que se encuentran más cercanos a p en el espacio de entrada.

  El método se fundamenta en la hipótesis de que patrones que se encuentren cercanos y, por tanto, tengan características similares es probable que pertenezcan a la misma clase. Como medida de la distancia entre los patrones se utiliza la distancia euclídea.

  Como desventaja de esta técnica podemos citar el alto consumo de memoria y el elevado tiempo de proceso necesario para clasificar nuevos patrones.

- **Random Forest (RF)**:

  Propuestos por primera vez por Ho (1995) y desarrollados, en 2001, por (Breiman, 2001; Hastie et al., 2003), los bosques aleatorios o Random Forest, agrupan un conjunto de Árboles de Decisión y hacen la clasificación final por un sistema de voto. Para realizar la agrupación RF utiliza un método llamado Bagging o Bootstrap Aggregating (Breiman, 1996). Las técnicas Bagging consisten en entrenar

diferentes clasificadores, cada uno con un subconjunto del conjunto de entrenamiento. Para crear dichos subconjuntos se usa una selección de patrones aleatoria, con reemplazamiento. La salida del clasificador global se obtiene combinando la salida de todos los clasificadores utilizando un sistema de voto (mayoría) o de pesos (suma ponderada de la salida de cada clasificador).

La combinación de la salida de varios clasificadores hace que esta técnica sea considerada más robusta frente al ruido y al sobreajuste, que los clasificadores individuales utilizados (Galar et al., 2012). Frente a los Árboles de Decisión, (Zhou, 2012) muestra que Random Forest es capaz de reducir simultáneamente el error y la varianza.

La implementación de Random Forest utilizada en los experimentos ha sido la incluida en la librería SciKit-Learn (*http://scikit-learn.org/stable/*).

2.4.3. Algoritmos de Regresión

Como técnicas de regresión se han utilizado Support Vector Machine para Regresión y Gradient Boosted Regression.

A continuación se explican brevemente dichas técnicas:

- **Support Vector Machine (SVM)** (Cortes and Vapnik, 1995):

  método de aprendizeje supervisado extensamente usado en problemas tanto de clasificación como regresión. Está basado en el uso de funciones *kernel* para contruir modelos no lineales, que generen hiperplanos de separación máxima entre las clases del problema. Diferentes funciones *kernel* puede ser utilizadas. Las más comunes son: lineales, polinomiales y de base radial (RBF). Información más detallada sobre SVM puede ser consultada en (Burges, 1998; Mandic and Goh, 2001).

  En este trabajo se ha hecho uso de la implementación que hace de SVM el paquete WEKA llamada Sequential Minimal Optimization (SMOreg) (Shevade et al., 1999; Smola and Schoelkopf, 1998) con funciones *kernel* lineal y RBF.

- **Gradient Boosted Regression (GBR)** (Friedman, 2001, 2002): combina Árboles de Regresión con Boosting (Schonlau et al., 2005). Este último es un método de agregación de modelos que permite aumentar el rendimiento predictivo.

Se basa en añadir secuencialmente nuevos modelos, cada uno de los cuales enfocado en resolver adecuadamente los errores cometidos por los modelos previos.

Se ha hecho uso del paquete Gbm (Ridgeway et al., 2013), incluido en R (Team, 2014).

Los motivos para elegir las técnicas de clasificación y regresión descritas han sido los siguientes:

K-NN es un método sencillo y eficaz, que presenta sensibilidad a la presencia de atributos irrelevantes. Por ello, resulta de interés para evaluar subconjuntos de atributos calculados con las técnicas de selección presentadas.

MLP es considerado un aproximador universal, siendo capaz de aprovechar relaciones no-lineales entre los atributos. Al contrario que J48, este clasificador es capaz de resolver problemas que no se ajusten necesariamente a una solución en forma de árbol.

J48 es una técnica muy potente, basada en una estructura de árbol y que hace uso de la teoría de la información. Presenta tolerancia a la presencia de atributos irrelevantes y/o redundantes. El estar basada en conceptos relativamente similares (teoría de la información) a las funciones de evaluación de las variantes de LIA, en general, y a LIA3 (árbol de decisión) en particular, debería garantizar la compatibilidad máxima de este clasificador con los métodos de selección de atributos desarrollados.

Random Forest aporta las ventajas de los árboles de decisión, unidas a una mayor robusted al ruido. Esta última característica, combinada con los buenos resultados que muestran las publicaciones en el ámbito del mercado de valores, haciendo uso de la misma, hacen que se haya considerado la técnica adecuada para realizar la experimentación con datos del mercado de valores.

En relación a los algoritmos de regresión, tanto **SVM** como **GBR** son dos reconocidas y potentes técnicas de regresión. En el caso del experimento de predicción de producción fotovoltaica se ha usado preferentemente GBR dado que parece ser más resistente al sobre-entrenamiento que SVM, al menos, en este problema.

3

Descripción de los métodos desarrollados

A lo largo de este capítulo se van a presentar nuevos métodos de selección de atributos y de clasificación.

Como métodos de selección de atributos, se han desarrollado: LIA (Selección de Atributos por Análisis Local de la Fiabilidad) y tres variantes que, basadas en las mismas funciones de evaluación, hacen uso de diferentes algoritmos de búsqueda de soluciones. Así, LIA2 implementa una búsqueda *greedy*. Por su parte, LIA3 combina una búsqueda *greedy* con una estructura en forma de árbol de decisión. Se ha desarrollado, también, una variante llamada LIAR, que es utilizable en problemas de selección de atributos para regresión.

Por último, se presenta un algoritmo de clasificación, CLIA, de aplicación específica a problemas de Tipo-I (Mercado de Valores).

Dado que todos los métodos desarrollados aquí se basan en los mismos fundamentos teóricos, se van a presentar los conceptos generales, antes de explicar, en detalle, cada método concreto.

3.1. Conceptos generales

Muchos métodos de aprendizaje, entre ellos los desarrollados en esta tesis, requieren de un procedimiento de búsqueda de soluciones y de un sistema para evaluar la bondad de cada solución. A lo largo de esta sección, se van a discutir algunas características

importantes de ambas tareas. También, se van a presentar algunas soluciones genéricas utilizadas en el desarrollo de las técnicas propuestas.

3.1.1. Riesgo verdadero *versus* riesgo empírico

En el capítulo de Estado de la Cuestión (ver Sección 2.2) se vió que el objetivo de un sistema de aprendizaje automático, es utilizar la información presente en el *conjunto de entrenamiento*, para desarrollar un modelo que pueda ser aplicado a datos desconocidos de forma que se optimize una medida dada.

Si los datos del *conjunto de entrenamiento* fueran exactamente iguales a los del *conjunto de test*, o, la distribución de ambos conjuntos fuera similar y contáramos con infinitos patrones, podríamos optimizar el modelo utilizando como entrada sencillamente el espacio de datos de entrada. Sin embargo, en el mundo real, se deben abordar dos problemas de cara a optimizar una medida dada. Por un lado, se debe afrontar el problema que viene dado por la diferencia entre el *conjunto de entrenamiento* y el *conjunto de test*. Así, si asumimos que ambos conjuntos son iguales, incurrimos en el llamado riesgo verdadero, también llamado error de generalización.

En la Ecuación 3.1 se define riesgo empírico como la probabilidad de error en la clasificación, en el *conjunto de entrenamiento*. X contiene el valor de todos los atributos. Y es un vector que contiene la clase para cada patrón de entrenamiento. En la Ecuación 3.2 se define riesgo real como la probabilidad de error en la clasificación, en el *conjunto de test*.

$$RiesgoEmpirico = Probabilidad(Clasificador(X) \neq Y) \tag{3.1}$$

$$RiesgoReal = Probabilidad(Clasificador(XTest) \neq YTest) \tag{3.2}$$

A continuación se muestra un ejemplo que ilustra en que riesgos se puede incurrir si para optimizar cierta medida, en este caso *precision*, ésta se optimiza simplemente sobre el *conjunto de entrenamiento* . Pensemos en un problema de clasificación, con dos clases (C_1 y C_0) donde la función que queremos optimizar es *precision* en patrones C_1. Dicha función se define en la Ecuación 3.3.

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

En la Figura 3.1 podemos observar la distribución de patrones, de ambas clases, proyectados sobre un mapa bidimensional, tanto para el *conjunto de entrenamiento* como para el *conjunto de test* .

Si observamos la proyección bidimensional correspondiente a los patrones del *conjunto de entrenamiento* (únicos patrones conocidos en fase de aprendizaje), se pueden delimitar tres posibles zonas interesantes: A, B y C. A contiene sólo 5 patrones, todos ellos rojos (*precision* del 100 %). La zona B, tiene 15 patrones, el 67 % rojos (67 % de *precision*) y la zona C, contiene 150 patrones, dando una *precision* del 57 %.

En Test, la zona A arroja una precisión del 0 %, la zona B del 64 % y la zona C del 53 %.

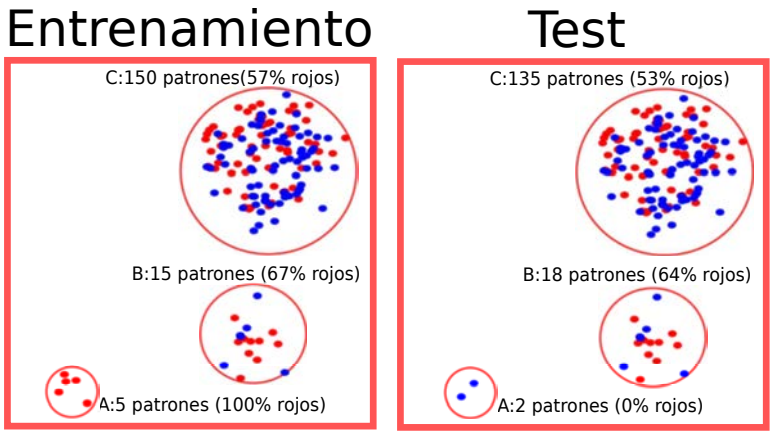

Figura 3.1: Entrenamiento *vs* Test

En la Tabla 3.1 se muestran el número de patrones, el error empírico y la *precision*, para cada zona, aplicada al *conjunto de entrenamiento* . Si analizamos los datos de dicha tabla, y quisieramos apostar por una zona que maximice *precision* (sobre el *conjunto de test*), podríamos pensar que la zona A, es un buen candidado, dado que arroja el valor más alto de *precision* (100 %). Sin embargo, dado que dicha zona contiene muy pocos patrones también podemos pensar que las zonas B y C, a pesar de aportar valores de *precision* menos elevados, sean más útiles de cara a aplicarse en el *conjunto de test*. Siguiendo este razonamiento intuitivo, llegaríamos a la conclusión que nuestro sistema de aprendizaje debe tener en cuenta no solo la función a optimizar si no, también la fiabilidad de la misma. Con ello, se predende minimizar, simultáneamente, el riesgo empírico y el riesgo real.

La la Tabla 3.2 muestra los mismos datos, en este caso, para el *conjunto de test*.

Zona	Patrones(C_1, C_0)	Riesgo empírico	*Precision*
A	(5 C_1/0 C_0)	0.00	5/5 = 100 %
B	(10 C_1/5 C_0)	0.33	10/15 = 67 %
C	(86 C_1/64 C_0)	0.46	86/150 = 57 %

Tabla 3.1: *Precision* en *conjunto de entrenamiento*

Zona	Patrones(C_1, C_0)	Riesgo real	*Precision*
A	(0 C_1/2 C_0)	1.00	0/2 = 0 %
B	(9 C_1/5 C_0)	0.36	9/14 = 64 %
C	(80 C_1/70 C_0)	0.47	80/150 = 53 %

Tabla 3.2: *Precision* en *conjunto de test*

En esta segunda tabla, se muestra el riesgo real frente al riesgo empírico mostrado en la anterior. Como podemos observar, los resultados para cada zona, varían respecto al *conjunto de entrenamiento*.

En la siguiente sección se va a presentar la solución técnica que se ha dado en esta tesis al problema aquí enunciado: riesgo empírico *vs* riesgo real. La idea que subyace y que es la base de todas las técnicas presentadas, consiste en formalizar un procedimiento tal que a partir de un *conjunto de entrenamiento*, posiblemente ruidoso y con escasez de patrones, sea capaz de generalizar soluciones que funcionen razonablemente bien sobre el *conjunto de test*. Para ello, resulta fundamental implementar un sistema capaz de estimar la fiabilidad de cada solución.

3.1.2. Funciones de evaluación y estimación de la fiabilidad

En esta tesis, se han desarrollado diferentes funciones de evaluación de soluciones, de forma que se maximicen diferentes medidas de calidad.

Se muestra a continuación la nomenclatura que se ha utilizado a lo largo de este capítulo, para la definición de funciones de evaluación y sus cálculos asociados:

- C_1 y C_0: C_1 es la clase objetivo, habitualmente la clase minoritaria. C_0 es la otra clase.
- *agrupacion*: conjunto de patrones, relacionados según algún criterio generalmente espacial (patrones cuyas proyecciones sobre mapas n-dimensionales, atendiendo

al valor de n atributos, pertenecen a una misma zona del espacio):

- $agrupacion.C_1$, $agrupacion.C_0$: número de patrones de clase C_1 y C_0, respectivamente, dentro de la agrupación.
- Si *total* representa a la agrupación de todos los patrones presentes en un problema, $total.C_1$ and $total.C_0$: serían el número de patrones C_1 y C_0 presentes en el problema.
- Si *celda* es una agrupación de patrones en una rejilla o estructura similar, $celda.C_1$ y $celda.C_0$ serían el número de patrones C_1 y C_0 en la celda.

- $densidad(agrupacion)$: densidad de patrones de la clase C_1 en $agrupacion$.

$$densidad(agrupacion) = \frac{agrupacion.C_1}{agrupacion.C_1 + agrupacion.C_0} \tag{3.4}$$

- $densidadTrivial$: densidad de patrones de la clase C_1 en el conjunto de datos completo.

$$densidadTrivial = densidad(total) = \frac{total.C_1}{total.C_1 + total.C_0} \tag{3.5}$$

- $confianza$: parámetro que modula la estimación de patrones fiables. A mayor valor, más conservadoras serán las estimaciones, descartando así una mayor cantidad de ruido en los datos y proporcionando soluciones que generalizan mejor.

  Valores excesivamente bajos provocarán que las soluciones no generalicen bien, por lo que el $riesgoreal >> riesgoempirico$. Por contra, valores excesivamente altos, provocarán que $riesgoreal \approx riesgoempirico$ pero la medida a optimizar (ej. *precision*) no alcance un valor óptimo. Por ello, es importante alcanzar un equilibrio entre generelización y maximización de la medida a optimizar.

- $dimension(problema)$: número de atributos relevantes para resolver óptimamente un problema dado.

- $dimension(solucion)$: número de atributos que contiene una solución dada.

Como se ha comentado en la Sección 3.1.1, en este mismo capítulo, para evaluar correctamente las zonas o agrupaciones de patrones, debemos tener en cuenta, además

del riesgo empírico, el riesgo real, es decir, debemos evaluar en base a la previsión de patrones C_1 y C_0 que, se estima, se mapean en dicha zona, en el conjunto de Test.

Así, la idea que subyace detrás de todas las funciones de evaluación desarrolladas, consiste en calcular, previamente a la función de evaluación, los patrones de cada clase que puedan estimarse como fiables. Para ello, se procede a calcular, para cada zona o agrupación de patrones a evaluar, la densidad de patrones de clase minoritaria, C_1 que puede ser esperable o predecible en base, solamente, a los efectos del azar, para un valor de confianza dado. Una vez eliminados dichos patrones, puede hacerse uso de la función concreta que queramos maximizar, función F, para evaluar cada zona. La estructura general que seguirán estas funciones de evaluación será entonces similar a la definida en el Algoritmo 1.

Algorithm 1 FUNCTION FEvalGenerica

Input:

celda: agrupación de patrones a evaluar.

$confianza$: valor de confianza.

total: agrupación que contiene todos los patrones del problema.

Output: Valor numérico real.

1: $densidadTrivial \leftarrow (total.C_1/(total.C_1 + total.C_0))$
2: $delta \leftarrow DeltaDensidad(celda.C_1, celda.C_0, confianza, densidadTrivial)$
3: **if** $delta > 0$ **then**
4: $\quad tc \leftarrow celda.C_0 + celda.C_1$
5: $\quad fiable.C_1 \leftarrow tc \times (densidadTrivial + delta)$
6: $\quad fiable.C_0 \leftarrow tc - fiableC_1$
7: $\quad$ **return** $F(fiable.C_1, fiable.C_0, total.C_1, total.C_0)$
8: **else**
9: $\quad$ **return** 0
10: **end if**

3.1.2.1. Definición de *DeltaDensidad*()

DeltaDensidad es una función que tiene como objetivo estimar cuantos patrones de clase C_1 en una celda son no explicables por desviaciones producto del azar. A los patrones no explicables por azar los denominaremos aquí patrones informados.

La función recibe como parámetros de entrada: una *celda* (agrupación de patrones, siendo $celda.C_1$ y $celda.C_0$: número de patrones C_1 y de C_0 en dicha agrupación. $confianza$: parámetro confianza y $total$: agrupación que contiene el total de patrones del problema.

Como salida, **DeltaDensidad** devuelve un número real en el intervalo: $[0, 1]$, que representa el incremento de densidad sobre la densidad trivial. El número de patrones informados, para un valor de confianza dado, podrá calcularse entonces con la Ecuación 3.6.

$$patronesInformados = deltaDensidad \times (celda.C_1 + celda.C_0) \tag{3.6}$$

También, podremos calcular (ver Ec. 3.7), el número de patrones de clase C_1 y C_0, que se estima obtendríamos en un conjunto de Test.

$$PatronesEsperablesC_1 = (densidadTrivial + deltaDensidad) \times (celda.C_1 + celda.C_0) \tag{3.7}$$

Para poder modelar hasta que punto un resultado se debe o no al azar, debemos introducir la idea de confianza. Así, un valor de confianza de un 95 % equivale a decir que existe un 5 % de probabilidades del evento contrario.

Teniendo en cuenta que esta técnica ha sido diseñada para problemas binarios el instrumento estadístico utilizado para modelar probabilidades ha sido la Distribución Binomial[1] que nos permite de una forma cómoda calcular probabilidades de sucesos para problemas de dos clases.

[1] El cálculo por medio de la Distribución Binomial es exacto, siempre y cuando, los sucesos aleatorios sean independientes. Si nuestro problema presenta autorrelación es conveniente realizar cambios en el cálculo de probabilidades.

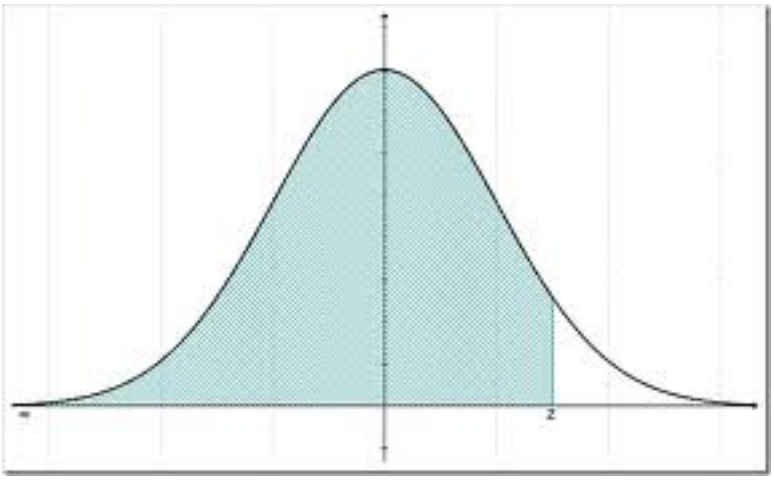

Figura 3.2: Distribución Binomial - Probabilidad acumulada

En 3.2 se muestra la probabilidad acumulada, zona sombreada, de que se produzca un número de eventos dados, menor que z. Así, para una probabilidad dada, p, podemos calcular cuantos eventos quedan a la izquierda de z, o lo que es lo mismo son esperables por puro azar, para dicha probabilidad p. Formalmente, diríamos que z eventos son esperables por azar, para un valor de confianza p, si $Probabilidad(X <= z) = p$. Podemos constatar como a mayor confianza menos eventos quedan a la derecha del punto de corte y, por tanto, menos eventos se podrán considerar informados o no producidos por azar.

Para hallar el valor de *DeltaDensidad* se procede a calcular, haciendo uso de la probabilidad acumulada en una distribucion binomial, 4 puntos, tales como los marcados en la Figura 3.3. NC_1 y NC_0, denotan el número de patrones C_1 y C_0, respectivamente, de la agrupación o celda a evaluar.

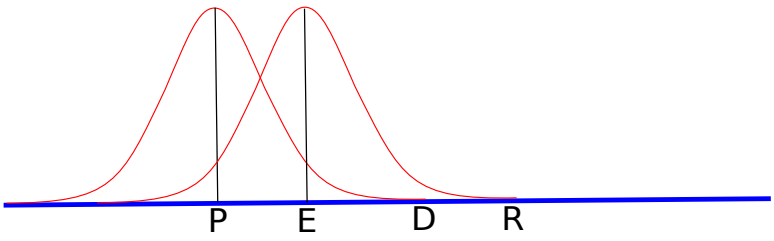

Figura 3.3: Cálculo del incremento de densidad fiable

Descripción de los puntos *P, E, D y R*:

- Punto *R*: $densidad = C_1/(C_1 + C_0)$
- Punto *P*: $densidadTrivial = total.C_1/(total.C_1 + total.C_0)$
- Punto *D*: desviación máxima calculada para una D. Binomial(C_1+C_0,$densidadTrivial$) y un valor de $confianza$.
- Punto *E*: densidad estimada, calculada como la densidad mínima que explique la obtención de una densidad *R*, con (C_1+C_0) patrones, para la desviación correspondiente a una probabilidad acumulada de $confianza$.

Es decir, la función *DeltaDensidad*, calcula la densidad mínima E, capaz de explicar la agrupación observada (C_1, C_0), haciendo uso de la estimación más pesimista de desviación repecto al promedio para una distribución binomial, de (C_1+C_0) patrones y $promedio = densidadTrivial$.

Si ($E > P$) consideramos que existe información porque la densidad mínima para generar los patrones de la celda es mayor que la densidad trivial. En ese caso, la función calcula $delta = E - P$. Por el contrario, en caso de que ($E <= P$) consideramos que dicha agrupación no aporta información fiable y la función calcula $delta = 0$. Una manera de visualizar este procedimiento es pensar en un filtro de valores anómalos funcionando a la inversa. Es decir, un filtro que deja pasar los valores anómalos y filtra los no anómalos. Para realizar el test de anomalía se usan, como herrmaientas matemáticas, los conceptos de intervalo de confianza y distribución binomial.

Una propiedad interesante que exhibe esta función consiste en que si contamos con infinitos patrones en nuestra agrupación, el punto *E* calculado coincidirá con el punto *R*. Esto resulta coherente con el hecho de que si contamos con infinitos patrones el punto *R* será totalmente fiable por lo que *E* debe coincidir con *R* en ese caso. Por contra, cuantos menos patrones formen nuestra agrupación más riesgo tendremos de error, por lo que el cálculo de densidad estimada será mucho más conservador y *E* será muy inferior a *R*.

Si bien, en estadística es común hablar de intervalo de confianza y éste tiene una relación muy estrecha con el concepto aquí utilizado, no debe nunca tomarse el valor del parámetro confianza de la técnica como intervalo de confianza matemáticamente exacto, ya que, existen varios factores (la distribución real de los datos y la autocorrelación entre

patrones, entre otros) que hacen que el cálculo sea en todo caso aproximado y, por ello, este parámetro debe entenderse como un medio para evaluar fiabilidades y no como un resultado estadístico formalmente preciso.

En el algoritmo 2 se formaliza la función **DeltaDensidad**. Dicha función implementa una búsqueda dicotómica del valor de delta que permita explicar el número de patrones C_1, para una desviación del promedio máxima, para un valor de confianza dada. La función CDFBINOM[1](i,p,n) calcula la probabilidad acumulada para un distribución binomial, de obtener i eventos, en una distribución binomial de parámetros (p, n).

A modo de ejemplo, pensemos en que tenemos un problema con 1000 patrones en el conjunto de Entrenamiento. 100 patrones de clase C_1 y 900 patrones de clase C_0. Así, $densidadTrivial = 0.10$.

Para calcular el valor de **DeltaDensidad** de una agrupación formada por 100 patrones, 12 de clase C_1 y 88 de clase C_0 ($R = 0.12$), y $confianza = 0.95$, procederíamos de la siguiente manera:

La probabilidad acumulada, $CDFBINOM(12; 0.10; 100) = 0.80$. Dado que $0.80 < 0.95$ podemos concluir que es posible que dicha agrupación se haya producido solo por azar. Así, el resultado de DeltaDensidad sería igual a 0.

Pensemos ahora en calcular la DeltaDensidad de una agrupación formada por 100 patrones, 20 de clase C_1 y 80 de clase C_0 ($R = 0.20$), y $conf = 0.95$, procederíamos de la siguiente manera: $CDFBINOM(20; 0.10; 100) = 0.9992$. Dado que $0.9992 > 0.95$ podemos concluir que, para ese nivel de confianza, existe información en la agrupación. Puede calcularse que $|CDFBINOM(20; 0.1452; 100) - 0.95| \approx 0$, por lo que se establece el punto $E = 0.1452$. Así, el resultado de DeltaDensidad sería $(P - E) = 0.1452 - 0.10 = 0.0452$. Lo que viene a equivaler a que en la agrupación de 20 patrones C_1 y 80 patrones C_0, podemos estimar que 5 patrones C_1 son informados. Se estima así, que usando esta misma zona sobre el conjunto de Test, obtendríamos 15 patrones de clase C_1 y 85 patrones de clase C_0.

Tomando como base el procedimiento aquí presentado, las funciones de evaluación de soluciones podrán realizar sus cálculos sobre los patrones estimados en cada zona y no, sobre los patrones realmente presentes en el conjunto de Entrenamiento en dicha

[1]Para calcular la probabilidad acumulada se ha hecho uso de la librería de calculo científico GSL. https://www.gnu.org/software/gsl

Algorithm 2 DeltaDensidad

Input:

NC_1, NC_0: número de patrones C_1 y C_0.

$confianza$: valor del parámetro confianza.

$densidadTrivial$: densidad trivial del problema.

Output: Valor numérico real.

$l1 \leftarrow densidadTrivial$
$p1 \leftarrow CDFBINOM(NC_1, l1, NC_1 + NC_0)$
if $p1 < confianza$ **then**
    return 0
end if
$l2 \leftarrow NC_1/(NC_1 + NC_0)$
$p2 \leftarrow CDFBINOM(NC_1, l2, NC_1 + NC_0)$
$\varepsilon \leftarrow \frac{1-confianza}{10}$)
$p \leftarrow -1$
while $|confianza - p| >= \varepsilon$ **do**
    $l \leftarrow \frac{l2-l1}{2} + l1$
    $p \leftarrow CDFBINOM(NC_1, l, NC_1 + NC_0)$
    if $confianza < p$ **then**
        $l1 \leftarrow l$
        $p1 \leftarrow p$
    else
        $l2 \leftarrow l$
        $p2 \leftarrow p$
    end if
end while
return $(l - densidadTrivial)$

zona. Con ello, se pretende que las decisiones que dichas funciones permitan tomar, se vean menos afectadas por la diferencia entre el riesgo empírico y el real, minimizando la diferencia entre ambos.

3.1.3. Monoticidad y búsquedas *greedy*

En este apartado, se va a tratar la monoticidad, propiedad presente el algunos problemas y que tiene importantes repercusiones en el tipo de búsqueda de soluciones que puede utilizarse.

Las Figuras 3.4, 3.5, 3.6 y 3.7 representan 4 problemas. Cada uno de ellos, consistente en problemas de clasificación de 2 clases: patrones de color rojo y azul. En dichos ejemplos, se trabaja con solo 2 atributos, y las figuras suponen la proyección de todos los patrones de entrenamiento, sobre un mapa bidimensional. Junto a cada mapa, se muestran unas marcas que representan histogramas de frecuencia de cada clase. Cada histograma consta de dos barras. La altura de cada barra representa el porcentaje de puntos de dicho color. Se muestran 4 histogramas por dimensión. Si pensamos que el rango total de cada dimensión es $[0, 1]$, cada histograma corresponderá a particiones de 0.25. Los histogramas horizontales corresponden a un atributo y los verticales a otro.

En el problema A, vemos que los patrones de cada clase se agrupan de forma no solapada con los de la otra. Los histogramas horizontales muestran una clara separación de cada clase. En una dimensión el histograma 1, primero por la izquierda, muestra un 100 % de patrones rojos, y los histogramas 3 y 4, 100 % azul. En vertical, podemos observar la misma situación: unos histogramas alcanzan 100 % de patrones rojos y otros azules. A partir de dichas observaciones, podemos decir que el problema A, presenta monoticidad. Esto permite que un método de búsqueda que funcione de forma incremental o *greedy*, será capaz de clasificar correctamente los patrones, empezando por detectar separaciones útiles entre estos, en una dimensión, ya sea X ó Y y, más adelante, refinar dichas zonas añadiendo la dimensión que falta.

En el problema B vemos que la distribución de patrones, por las simetrías que presenta, genera unos histogramas de frecuencia, prácticamente similares para ambos atributos, y para todas las particiones. Todos los histogramas rondan el 50 % de patrones de cada clase. Así, un método que trabajara en una dimensión, no sería capaz de detectar particiones ventajosas de los patrones.

El problema C, muestra una situación donde ambas clases están solapadas pero existe una zona donde claramente dominan los patrones rojos. Además, el problema no presenta simetrías unidimensionales importantes. Como puede observarse, en los histogramas horizontal 1 y vertical 1, existen indicios en cada dimensión, que permitirían a un método incremental dado, encontrar la zona de máxima densidad de patrones rojos.

El problema D presenta una distribución de patrones muy solapada. Aún así, vemos que en la esquina superior izquierda existe una zona de una densidad alta de patrones rojos. Sin embargo, como podemos observar en los histogramas unidimensionales, vemos que para un método incremental sería muy difícil detectar en una dimensión cualquiera de las dos particiones que finalmente definen la zona de máxima densidad de rojos.

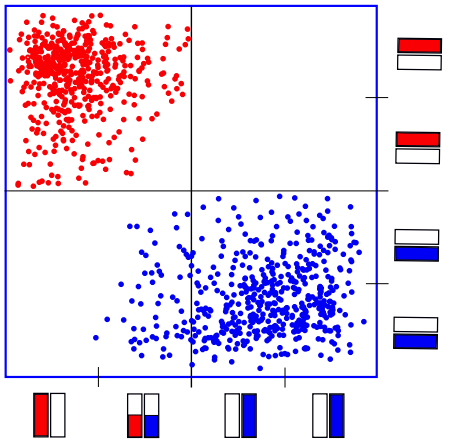

Figura 3.4: Problema A

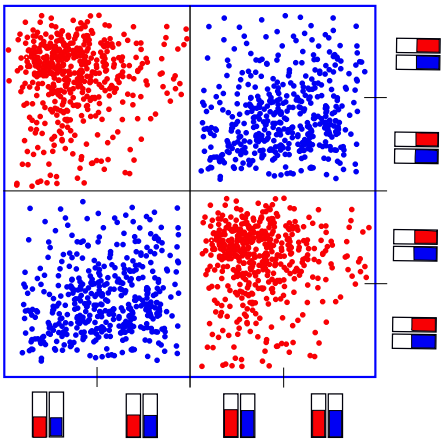

Figura 3.5: Problema B

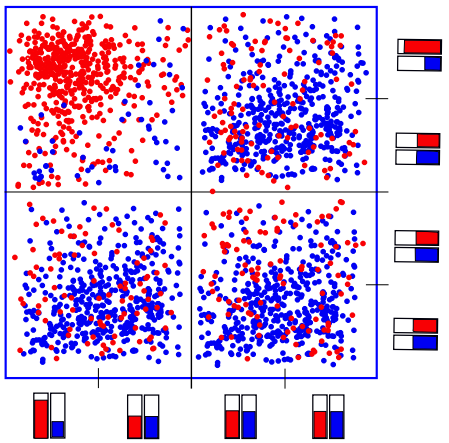

Figura 3.6: Problema C

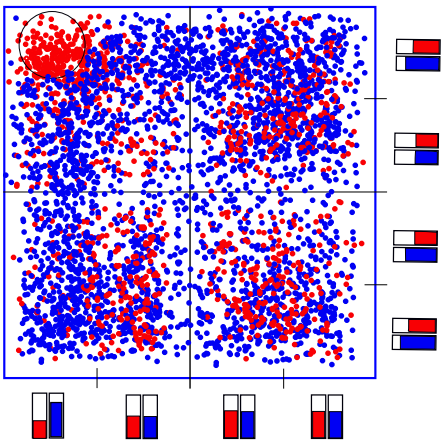

Figura 3.7: Problema D

De esta forma, los problemas A y C, podrán ser resueltos ventajosamente por algoritmos que avancen incrementalmente hacia la solución completa. Dichos algoritmos,

presentan importantes ventajas computacionales ya que exploran el espacio de soluciones de una en una dimensión.

Por contra, los problemas B y D, solo van a ser resueltos óptimamente, por algoritmos que exploren directamente en dimensiones mayores a 1. Así, un algoritmo que evalue cada partición 2D, encontrará fácilmente las zonas de máxima densidad de patrones rojos. Como inconveniente de estos últimos algoritmos aparece, en primer lugar, la gran demanda computacional requerida. Este requisito hace que estas aproximaciones exhaustivas escalen mal a problemas con un gran número de atributos. Por poner un ejemplo de la carga computacional, pensemos en la exploración, en 3 dimensiones y 8 particiones por dimensión. Un algoritmo exhaustivo deberá evaluar $8 \times 8 \times 8 = 512$ zonas. Frente a ésto, un algoritmo greedy, le bastará con evaluar $8 \times 3 = 24$ zonas.

3.1.4. Búsqueda de regiones objetivo: particionado por rejilla

En las secciones previas se han presentado funciones de evaluación capaces de estimar la bondad de cada agrupación de patrones. Para completar el método, se presenta aquí el procedimiento general de búsqueda de esas agrupaciones de patrones. Cada una de las técnicas desarrolladas utiliza, con ligeras variaciones, el procedimiento de búsqueda de regiones objetivo, ésto es, de regiones en el espacio de entrada donde la densidad de patrones de la clase minoritaria sea especialmente alta, una estructura de datos que representa la distribución espacial de éstos, llamada rejilla.

En la Figura 3.8 se muestra, a modo de ejemplo, el funcionamiento del particionado por rejilla:

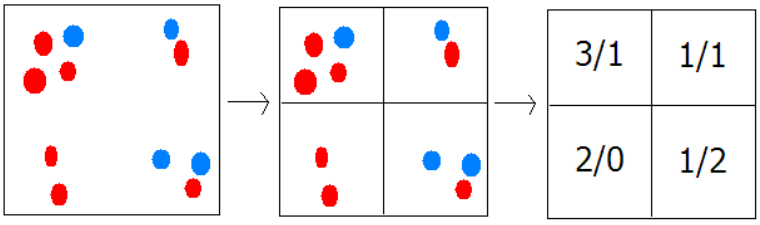

Figura 3.8: Particionado por rejilla

En el cuadro primero vemos un conjunto de patrones, rojos y azules distribuidos por el espacio de entrada según los valores de 2 de sus atributos. Si dividimos dicho espacio en regiones podemos contabilizar cúantos patrones de cada clase caen dentro

de cada división, llamadas aquí celdas. Una vez obtenidas las celdas, podemos evaluar también cualquier hipercelda (toda región rectangular que pueda inscribirse en la rejilla) simplemente sumando los contadores de patrones de cada clase que aporta cada celda al conjunto total. En el ejemplo expuesto podríamos construir una hipercelda que abarcara las dos celdas de la izquierda. Así, $Hipercelda.C_1 = 3 + 2 = 5$ y $hipercelda.C_0 = 1 + 0 = 1$.

3.1.5. Procedimiento general de búsqueda

El procedimiento general de búsqueda de regiones se resume del siquiente modo:

- Dividir el espacio de entrada en regiones cuadradas y de tamaño fijo, llamadas celdas, que en conjunto forman una rejilla. El número de dimensiones con que trabaja el algoritmo depende de cada implementación concreta ya sea exhaustiva o *greedy* y de cada etapa de procesamiento.
- Mapear cada patrón de datos en la celda correspondiente, en base al valor de los atributos seleccionados, contabilizando para cada celda el número de patrones de cada clase.
- Agrupar dichas celdas en hiperceldas.
- Haciendo uso de una función de evaluación, se puede entonces seleccionar la hipercelda óptima.

3.1.6. Ventajas del particionado por rejilla

El particionado por rejilla evalúa agrupaciones de patrones atendiendo a su cercanía en el espacio de entrada. Muchos otros métodos de selección de atributos y de clasificación, realizan esa misma agrupación basándose, a menudo, en el concepto de distancia entre patrones. Ejemplos de ello los encontramos en: K-NN, K-Means y muchos otros.

Ventajas del particionado por rejilla frente a estos otros basados en distancias:

1. Calcular las distancias entre un patrón y todos los demás, requiere un algoritmo de orden NxN. Frente a ésto, el particionado por rejilla es capaz de agrupar los patrones usando un algoritmo de orden N.

2. Implementación: el particionado por rejilla es capaz de agrupar N patrones en una rejilla de un número arbitrario de dimensiones, simplemente realizando operaciones matemáticas sencillas, con números enteros y sin uso de condicionales[1]. Para mapear un patrón de datos en base al valor normalizado de na atributos, en una rejilla de na dimensiones y P particiones para cada dimensión podemos calcular la coordenada que corresponderá a dicho patrón en un vector que almacenará la rejilla como valores contiguos haciendo uso de la fórmula:

$$indice = \sum_{i=1}^{na} patronEntrada[i] * P^i \tag{3.8}$$

   De esta forma, al agrupar todos los patrones que se mapean a una misma celda o hipercelda de una rejilla, estamos realmente evaluando patrones que se encuentran espacialmente cercanos sin la desventaja de tener que comparar todos los patrones entre sí. Una vez computada una rejilla, los procesos posteriores operan con un número reducido de celdas, y no con los N patrones iniciales. Ésto dota de eficacia suficiente a todos los procesos posteriores basados en éstas.

3. Reducción de la dimensionalidad de una rejilla: la versatilidad que presenta esta estructura de datos permite realizar, eficientemente, algunas operaciones comunes. Por ejemplo, si disponemos de una rejilla 3D donde están mapeados todos los patrones de nuestro problema, para obtener una rejilla 2D solo tenemos que sumar las celdas pertinentes. De esta forma, los cálculos se reducen a procesar cada celda de la rejilla origen frente a volver a recorrer todo el conjunto de patrones de Entremiento que sería necesario para crear de la nada dicha rejilla.

$$celda2D[x, y] = \sum_{z=1}^{P} celda3D[x, y, z] \tag{3.9}$$

   Por ejemplo, para un problema con 100.000 patrones de datos, computar una rejilla 2D, requerirá 100.000 operaciones. Frente a ésto, redimensionar una rejilla3D, con $P = 8$, en otra 2D solo requerirá $8^3 = 512$ operaciones.

[1] Las condicionales degradan el rendimiento de los *pipelines* de las CPU modernas.

3.2. Pretratamiento: Discretización Supervisada

Todas las técnicas desarrolladas en esta tesis, se enfocan a procesar atributos numéricos y discretizados en forma de números enteros. Por ello, antes de procesar cada problema, las técnicas aquí presentadas utilizan el mismo algoritmo de discretización. Los objetivos de este algoritmo son primeramente convertir los valores de los diferentes atributos a una forma tratable por el resto de técnicas y, secundariamente, esta discretización se realiza de forma que los datos codificados retengan la mayor parte de la información presente en los datos originales. Para alcanzar estos objetivos se ha diseñado un algoritmo de discretización encuadrado dentro de la categoría de algoritmos de discretización supervisada. Esto es, el algoritmo hace uso de la clase de salida para elegir la codificación más adecuada para cada atributo.

En la Tabla 3.3 se muestra cómo se codifican los valores de cada atributo atendiendo a su tipo.

Tabla 3.3: Discretización Atributos

Tipo Atributo	Valores	Codificado
Constante	{c}	{1}
Binario	{0,1}	{0,1}
Categórico	$\{c_1,c_2,...,c_n\}$, c_i cualquier texto	{0,1,...,n}
Número Real	$\{r_1,r_2,...,r_n\}$, $r_i \in \mathbb{R}$	{0,1,...,7}

Para la discretización de los valores compuestos por números reales se utiliza el Algoritmo 3, **DiscretizaReal**. El algoritmo intenta discretizar el atributo utilizando intervalos de ancho constante. Si el número de patrones que se discretizan en cada intervalo es más o menos uniforme para todos ellos, se discretiza usando dichos intervalos. Por el contrario, si la distribución de los datos provoca que el número de patrones de cada intervalo no sea uniforme, se calculan los intervalos de discretización de forma que el número de patrones que corresponde a cada uno sea constante. Una vez calculados los histogramas correspondientes a cada intervalo de discretización: ancho constante y alto constante, se compara la información que proporciona cada una de las dos codificaciones. La más informada es la finalmente utilizada para discretizar el atributo. Empíricamente se ha decidido que el número de intervalos de discretización para números reales sea 8.

El método **transform** discretiza cada atributo de forma que cada valor real es sustituido por un número entero correspondiente al índice del intervalo al que pertenece el valor real. Si llamamos *intervalos* a la lista de intervalos de discretización e *ini* y fin a los valores que indican el rango inicial y final de un intervalo. El valor real v será discretizado al valor i si $intervalos_i.ini \geq v \wedge intervalos_i.fin \leq v$.

La función **IsUniforme** devuelve $True$ si el número de patrones que son discretizados a cada intervalo es aproximadamente igual ($\pm 15\,\%$) para todos los intervalos.

La función **InformacionHist** devuelve un número real que contabiliza la información (entendida aquí como el total de patrones de clase C_1 no explicables por azar) contenida en los intervalos de discretización del histograma recibido como parámetro. El cálculo de la desviación respecto al promedio se realiza, en este caso, utilizando la distribución Normal (ver Ec. 3.10).

$$DesviacionTipicaDistNormal(n,p) = \sqrt{n \times p \times (1-p)} \tag{3.10}$$

Algorithm 3 DiscretizaReal

Input:

valores es un vector que contiene cada valor que toma el atributo a discretizar para cada patrón de datos.

nIntervalos: número de intervalos en los que discretizar cada valor.

Output: $codificado$ es un vector que contendrá los valores discretizados.

1: $hist_1 \leftarrow histogramaWidthConstante(valores, nIntervalos)$
2: **if** $isUniforme(hist_1)$ **then**
3: $\quad codificado_i = hist_1.transform(valor_i)$
4: **else**
5: $\quad hist_2 \leftarrow histogramaHeightConstante(valores, nIntervalos)$
6: $\quad$ **if** $informacionHist(hist_1) > informacionHist(hist_2)$ **then**
7: $\quad\quad codificado_i = hist_1.transform(valor_i)$
8: $\quad$ **else**
9: $\quad\quad codificado_i = hist_2.transform(valor_i)$
10: $\quad$ **end if**
11: **end if**
12: **return** $codificado$

Algorithm 3 DiscretizaReal (continuado)

13: **function IsUniforme**

Input:

hist es un histograma.

Output: Devuelte $True$ si todos los intervalos contienen, aproximadamente, el mismo número de patrones.

14: $minimo \leftarrow \frac{hist.patrones}{hist.nIntervalos} \times 0.85$
15: $maximo \leftarrow \frac{hist.patrones}{hist.nIntervalos} \times 1.15$
16: $uniforme \leftarrow 0$
17: **for** i in [1, hist.nIntervalos] **do**
18: **if** $minimo <= hist.intervalos(i).nPatrones >= maximo$ **then**
19: $uniforme \leftarrow uniforme + 1$
20: **end if**
21: **end for**
22: **if** $uniforme = hist.nIntervalos$ **then**
23: **return** $True$
24: **else**
25: **return** $False$
26: **end if**
27: **end function**

Algorithm 3 DiscretizaReal (continuado)

28: **function InformacionHist**

Input:

hist es un histograma.

Output: Devuelve un valor numérico real, que representa la información contenida en el total de los intervalos del histograma.

29: $trivial \leftarrow \frac{hist.patrones}{hist.patronesC_1}$
30: $informacion \leftarrow 0$
31: **for** i in [1, hist.nIntervalos] **do**
32: $n \leftarrow hist.intervalos(i).patrones$
33: $esperados \leftarrow (n \times trivial) + \sqrt{n \times trivial \times (1.0 - trivial)}$
34: $infoIntervalo \leftarrow hist.intervalos(i).C_1 - esperados$
35: **if** $infoIntervalo >= 0.0$ **then**
36: $informacion \leftarrow informacion + infoIntervalo$
37: **end if**
38: **end for**
39: **return** $informacion$
40: **end function**

3.3. LIA: Selección de Atributos por Análisis Local de la Información Fiable

En numerosos campos de aplicación del Aprendizaje Automático la presencia de ciertos factores (ruido, atributos irrelevantes, bajos ratios de información versus ruido, desbalanceo de clases y escasez de patrones), aumenta la complejidad del análisis y el rendimiento de algunas técnicas comunes se podría ver degradado. LIA, se ha diseñado específicamente para ser robusta frente a estos factores.

LIA pertenece a la categoría de *Filters* (selección de atributos independiente del clasificador) y, específicamente, a *Rankers* (generan una lista de atributos ordenada por relevancia). El algoritmo propuesto se fundamenta en la idea de descartar todo el ruido posible antes de realizar cómputos sobre la correlación de los atributos y la clase de salida. Dicho descarte de ruido se realiza localmente para cada subconjunto de atributos y rangos de valores analizados. Una vez descartada la información sospechosa de ser

debida al azar, la información resultante puede considerarse fiable (para un valor de confianza dado) y, por tanto, las correlaciones de atributos y rangos de valores de éstos, con la clase de salida pueden ser correctamente analizados. Una vez descartados los patrones considerados no fiables, puede hacerse uso de diferentes funciones, diseñadas para optimizar uno u otro objetivo.

La técnica permite elegir, por medio del parámetro *mode*, el criterio que se desea optimizar. Para $mode = 1$, el algoritmo devolverá una lista de atributos ordenados por su capacidad para definir regiones del espacio con alta densidad de patrones de la clase C_1. Para $mode = 2$, la lista de atributos se obtendrá ordenando éstos por la ganancia de entropía que puede proporcionar cada atributo.

LIA hace uso de una búsqueda exhaustiva [1], con la finalidad de poder enfrentar con éxito problemas donde existe poca o nula monoticidad (ver Sección 3.1.3), pero limitando la búsqueda a tres dimensiones. La limitación a tres dimensiones viene motivada por: 1. El tiempo de proceso para mayores profundidades podría ser inasumible para problemas de unos pocos cientos de atributos. 2. Experimentalmente no se ha constatado la necesidad de emplear mayores profundidades para obtener buenos resultados. 3. El número de patrones de entrenamiento disponibles habitualmente es demasiado escaso para rejillas de $dimensiones > 3$. Por ejemplo, 25.000 patrones repartidos entre las celdas de una rejilla de dimensión 4, para 8 particiones por dimensión, darían lugar a $(25.000 \div 8^4 = 6)$ patrones por celda. Seis patrones es un número insuficiente para poder sacar conclusiones estadísticamente robustas.

Si unimos esta decisión con una implementación eficiente, el diseño resultante será aplicable en problemas de tamaño medio y donde existan los factores de dificultad enumerados.

A continuación se describen las funciones de evaluación utilizadas en LIA, el algoritmo detallado, y unas notas sobre implementación y usabilidad.

[1] Cover y Van Campenhout (Cover and Van Campenhout, 1977) demostraron que, el único procedimiento de selección de atributos que puede garantizar la obtención del subconjunto óptimo, es la búsqueda exhaustiva. Desafortunadamente, la búsqueda exhaustiva es un problema NP-Completo(Davies and Russell, 1994).

3.3.1. Funciones de evaluación: $\mathbf{F_1}$ y $\mathbf{F_2}$

Para evaluar cada celda, agrupación de patrones de clases C_1 y C_0, el algoritmo LIA hace uso de las funciones $\mathbf{F_1}$ y $\mathbf{F_2}$, usando una u otra, dependiendo del valor del parámetro *mode*. En la definición de las mismas se hace uso de la nomenclatura presentada en la Sección 3.1.2.

Ambas funciones siguen el esquema general presentado en el Algoritmo 1. Así, la idea que subyace detrás de estas funciones de evaluación, consiste en calcular, para cada zona o agrupación de patrones a evaluar, la densidad de patrones de clase C_1, que puede ser esperable o predecible en base, solamente, a los efectos del azar, para un valor de confianza dado. Una vez corregidos dichos patrones, puede hacerse uso de la función concreta que queramos maximizar para evaluar cada zona.

Para calcular el número de patrones C_1, considerados fiables, dentro de la agrupación *celda*, se hace uso de la función **DeltaDensidad**, (ver Alg. 2). Ésta calcula el incremento de densidad de patrones de clase C_1 sobre la densidad global, para una celda dada.

En el Algoritmo 4 se describe la función $\mathbf{F_1}$. Dicha función evalúa una celda, en base a un valor de confianza dado, y devuelve la estimación de *precision* (ver Ec. 2.3). La función primero procede calculando los patrones de clase C_1, considerados fiables, y ya sobre estos realiza el cálculo de la *precision*.

Algorithm 4 FUNCTION $\mathbf{F_1}$

Input:

celda: una celda de el grid.

$confianza$: valor de confianza.

total: agrupación que contiene todos los patrones.

Output: La función devuelve la precisión fiable estimada.

1: $densidadTrivial \leftarrow total.C_1 \div (total.C_1 + total.C_0)$
2: $\delta densidad \leftarrow DeltaDensidad(celda.C_1, celda.C_0, confianza, densidadTrivial)$
3: $nt \leftarrow celda.C_1 + celda.C_0$
4: $fiable.C_1 \leftarrow nt * (densidadTrivial + \delta densidad)$
5: $fiable.C_0 \leftarrow nt - fiable.C_1$
6: **return** $fiable.C_1 \div (fiable.C_1 + fiable.C_0)$

En el Algoritmo 5 se formaliza el algoritmo utilizado para definir la función **F2**.

Dicha función calcula la ganancia de información que obtendríamos si dividimos el conjunto global de patrones, *total*, en dos subconjuntos: *celda* y *resto*. Este último contendrá los patrones no pertenecientes a la agrupación *celda*. Tanto *celda* como *total* representan agrupaciones de patrones y contienen, cada una, dos campos: C_1 y C_0. Cada uno contabiliza el número de patrones de cada clase presentes en la agrupación. Antes de realizar otros cálculos la función calcula cuantos patrones de clase C_1 son fiables. Una vez calculados los patrones C_1 considerados fiables, para un nivel de confianza dado, la función hace uso del concepto de ganancia de información o *Information Gain*, Ecuación 3.11.

$$IG(T, a) = H(T) - H(T|a) \tag{3.11}$$

La ganancia de información de un conjunto T y un criterio a, se define como la diferencia de la entropía del conjunto T menos la resultante de aplicar el criterio a al conjunto T. Así, la ganancia de información resultante de aplicar como criterio de división, que un patrón se encuentre dentro o fuera de una celda dada, se computa, Ecuación 3.12, calculando la diferencia entre la entropía del conjunto global de patrones, *total*, y la que se obtiene dividiendo éste en dos subconjuntos, patrones dentro y fuera de *celda*.

$$IG(total, celda) = H(total) - (H(celda) + H(total - celda)) \tag{3.12}$$

Dado que lo ideal es calcular la ganancia de información real, que proporciona una celda, pero solo disponemos de un conjunto limitado de Entrenamiento, la función **F2**, en aplicación del concepto de fiabilidad, solo tiene en cuenta como patrones de clase C_1 pertenecientes a *celda*, los patrones que se consideren fiables. El resto de patrones de clase C_1, que en Entrenamiento, pertenecen a *celda* se estima que no es probable que aparezcan en Test, por lo que la función los desecha antes de calcular IG.

Para calcular la entropía de una agrupación se usa la función F_{Ent}, defida en la Ecuación 3.13. F_{Ent} calcula la información (medida en promedio de número de bits) necesaria para codificar cada patrón, en una agrupación de patrones de clase C_1 y C_0. *Agr* representa a la agrupación de patrones y contiene dos campos: C_1 y C_0. Cada uno

contabiliza el número de patrones de cada clase presente en la agrupación. $Agr.Total$ se define como $(Agr.C_1 + Agr.C_0)$.

$$F_{Ent}(Agr) = -(\frac{Agr.C1}{Agr.Total} \times \log(\frac{Agr.C1}{Agr.Total}) + \frac{Agr.C0}{Agr.Total} \times \log(\frac{Agr.C0}{Agr.Total})) \quad (3.13)$$

Algorithm 5 FUNCTION $\mathbf{F_2}$

Input:

$celda$: una celda de el grid.

$confianza$: valor de confianza.

$total$: agrupación que contiene todos los patrones.

Output: La función devuelve la precisión fiable estimada.

1: $densidadTrivial \leftarrow total.C_1 \div (total.C_1 + total.C_0)$
2: $\delta densidad \leftarrow DeltaDensidad(celda.C_1, celda.C_0, confianza, densidadTrivial)$
3: $nt \leftarrow celda.C_1 + celda.C_0$
4: $fiable.C_1 \leftarrow nt * (densidadTrivial + \delta densidad)$
5: $fiable.C_0 \leftarrow nt - fiable.C_1$
6: $resto.C_0 \leftarrow total.C_0 - fiable.C_0$
7: $resto.C_1 \leftarrow total.C_1 - fiable.C_1$
8: $tr \leftarrow fiable.C_1 + fiable.C_0$
9: $tt \leftarrow total.C_1 + total.C_0$
10: $r \leftarrow tr/tt$
11: **return** $F_{Ent}(total) - (r \times F_{Ent}(fiable) + (1 - r) \times F_{Ent}(resto))$

3.3.2. Algoritmo

LIA se define en el Algoritmo 6. Se basa en evaluar la información aportada por todas las combinaciones de *depth* atributos. Para ello, se mapean los patrones, en *grids* o rejillas, de dimensión *depth*, usando el valor de cada atributo.

Para cada rejilla, el algoritmo calcula la celda que provee de mayor información, haciendo uso de la función la función **ComputeInfoBestCell**, Algoritmo 7. Esta función devuelve un vector que se almacena en $VInfoMax$ que contiene el resultado de evaluar la mejor celda de la rejilla, para cada valor de confianza dado. Para localizar la mejor celda de una rejilla, la función calcula la celda que aporta mayor información para todos los niveles de confianza considerados.

Para calcular la información de cada celda, se hace uso de la función $\mathbf{F_1}$ (ver Alg. 4) o de la función $\mathbf{F_2}$ (ver Alg. 5). Se usará una u otra dependiendo del valor del parámetro *mode*. Si $mode = 1$, se hará uso de $\mathbf{F_1}$. Para $mode = 2$, se usará $\mathbf{F_2}$. La configuración $mode = 1$ se usará con el objetivo de encontrar una lista de atributos, ordenados por mayor a menor capacidad, para definir zonas donde la densidad de patrones de clase C_1 se haga máxima. $mode = 2$, configura la técnica con el objetivo de encontrar la lista de atributos relevantes siendo éstos los que aporten globalmente mayor información al problema. En este contexto, se entiende información como la reducción de entropía.

A continuación, se acumula dicha información, para cada uno de los atributos que forman la combinación, en la matrix MI. MI tendrá tantas filas como posibles valores de confianza. Cada fila en MI almacena $nAtrs$ valores, correspondientes a la evaluación de cada atributo para un valor de confianza dado. Si $mode = 1$, la matriz MI almacenará para cada atributo y valor de confianza, el valor máximo evaluado para las combinaciones donde ha tomado parte dicho atributo. Si $mode = 2$, MI almacena el valor acumulado de información para el atributo y valor de confianza.

Por último, la lista final de atributos se obtiene computando el sumatorio de las filas normalizadas de dicha matriz, utilizando la función **SumatorioFilasNormalizadas**, Algoritmo 8. VR contendrá entonces un vector de tantas componentes como atributos tenga nuestro problema, y cada elemento tendrá un valor que representa la relevancia de cada atributo. Si ordenamos éstos, por dichos valores, obtendremos la lista de atributos más relevantes.

Una posible optimización que puede mejorar el tiempo de proceso para $depth = 3$, manteniendo un alto grado de precisión, consiste en calcular primero para $depth = 2$. Posteriormente, repetir el proceso con $depth = 3$ pero solo para combinaciones de atributos donde al menos dos de ellos reporten información en $depth = 2$.

La complejidad del algoritmo es $O(N)$ para evaluar cada combinación de atributos, donde N es el número de patrones. La complejidad total es $O(\binom{nAtrs}{depth} \times N)$, siendo *depth* el tamaño de los subconjutos de atributos evaluados. En el caso más desfavorable, para $depth = 3$, la complejidad será $O(\binom{nAtrs}{3} \times N)$, si bien tenderá a $O(\binom{nAtrs}{2} \times N)$ si se aplica la optimización enunciada.

Algorithm 6 LIA

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo. Por defecto: 8.

depth: profundidad de exploración.

CP_{min},CP_{max}: valores de confianza mínimo y máximo.

CP_{step}: tamaño del paso.

mode: establece el modo de funcionamiento. Puede valer 1 ó 2. Por defecto, 2.

Output: VR: vector con $nAtrs$ elementos. El elemento $VR[i]$ contiene la relevancia del atributo i.

1: $C \leftarrow 0$
2: **for** *cp in* $[CP_{min}..CP_{max}$, *step* $CP_{step}]$ **do**
3: $\quad VConf[C] \leftarrow 1 - 10^{-cp}$
4: $\quad C \leftarrow C + 1$
5: **end for**
6: $MI \leftarrow$ *new Array(nAtrs,C)*
7: **for** $(a_1, .., a_n)$ in $[combination(nAtrs, depth)]$ **do**
8: $\quad grid \leftarrow$ *new Grid(P,depth)*
9: $\quad grid.rellena(X, Y, , a_1, .., a_n)$
10: $\quad$ **if** $mode = 1$ **then**
11: $\quad\quad VInfoMax \leftarrow ComputeInfoBestCell(grid, VConf, F1)$
12: $\quad$ **else**
13: $\quad\quad VInfoMax \leftarrow ComputeInfoBestCell(grid, VConf, F2)$
14: $\quad$ **end if**
15: $\quad$ **for** i in $[1..C]$ **do**
16: $\quad\quad$ **for** j in $[1..n]$ **do**
17: $\quad\quad\quad$ **if** $mode = 1$ **then**
18: $\quad\quad\quad\quad MI[a_j, i] \leftarrow Max(MI[a_j, i], VInfoMax[i])$
19: $\quad\quad\quad$ **else**
20: $\quad\quad\quad\quad MI[a_j, i] \leftarrow MI[a_j, i] + VInfoMax[i]$
21: $\quad\quad\quad$ **end if**
22: $\quad\quad$ **end for**
23: $\quad$ **end for**
24: **end for**
25: $MR \leftarrow SumatorioFilasNormalizadas(MI, C, nAtrs)$
26: **return** MR

Algorithm 7 FUNCTION **ComputeInfoBestCell**

Input:

grid: estructura que contiene todas las celdas de una rejilla.

$VConf$: vector que contiene una lista de valores de confianza.

F: función utilizada para evaluar cada celda.

Output: $VInfoMax$ vector retornado por la función, con C componentes.

```
1: C ← |VConf|
2: for cell in [grid] do
3:     SumVInfo ← 0
4:     for i in [1..C] do
5:         VInfo[i] ← F(cell, VConf[i], total)
6:         SumVInfoMax ← SumVInfoMax + VInfo[i]
7:     end for
8:     if SumVInfo > SumVInfoMax then
9:         for i in [1..C] do
10:            VInfoMax[i] ← VInfo[i]
11:        end for
12:        SumVInfoMax ← ∑ VInfoMax
13:    end if
14: end for
15: return VInfoMax
```

Algorithm 8 FUNCTION **SumatorioFilasNormalizadas**

Input:

MI: Matriz de $filas$ filas x $cols$ columnas. Cada elemento $[i, j]$ contiene la información calculada para el atributo j, para el valor de confianza i.

$cols$: número de columnas.

$filas$: número de filas.

Output: VR vector retornado, con $cols$ elementos.

```
1: for i in [1..filas] do
2:     maximun ← 0
3:     for j in [1..cols] do
4:         if MI[i, j] > maximun then
5:             maximun ← MI[i, j]
6:         end if
7:     end for
8:     for j in [1..cols] do
9:         VR[j] ← VR[j] + (MI[i, j] ÷ maximum)
10:    end for
11: end for
12: return VR
```

Este diseño se ha pensado para enfrentarse a problemas donde un número relativamente pequeño de atributos relevantes se encuentran embebidos entre cientos o miles de atributos irrelevantes. Además, el algoritmo, para $depth > 1$, es capaz de descubrir información relevante en problemas donde exista poca o ninguna monoticidad a dimensiones bajas (ver Sección 3.1.3).

3.3.3. Eficiencia de la implementación

LIA sacrifica la escalabilidad a problemas con un gran número de atributos en aras de implementar una búsqueda exhaustiva que permita obtener la máxima precisión. Si bien todo algoritmo de orden $O(N^2)$, o superior, tiene unas limitaciones importantes en lo que a escalabilidad se refiere, es posible, gracias a una implementación inteligente, ampliar el rango de problemas donde LIA será utilizable a problemas de hasta varios miles de atributos.

Lo que posibilita esta implementación eficaz es el hecho de que el algoritmo consiste

en cálculos repetitivos, sencillos y sin dependencias temporales entre ellos. En base a esto, nada impide utilizar técnicas de proceso masivamente paralelo para procesar la aportación de cada rejilla al cálculo final. Esto supone una ventaja fundamental sobre otros métodos que, por presentar dependencias entre unos cálculos y los siguientes, por ejemplo, entre los distintos niveles de los árboles de decisión, tienen más dificultades para hacer uso del paralelismo.

Las técnicas, a nivel de implementación, que se han utilizado en LIA para acelerar en lo posible el algoritmo han sido:

1. Multihread. Una de las técnicas disponibles para lograr que un algoritmo haga un uso más completo de los recursos hardware con que cuentan las CPU modernas consiste en la implementación multithread[1]. Gracias a esta técnica, podemos crear nt hilos ($threads$), siendo responsable cada hilo de procesar nr rejillas, siendo $nr = TotalRejillas/nt$. Dado que no existen dependencias entre los cálculos, no es necesario compartir información durante el proceso, por lo que podemos procesar en paralelo todas las rejillas necesarias y, al final del cálculo, combinar la aportación de cada atributo al cálculo global. Por ejemplo, creando 8 hilos[2] en el bucle principal del programa, se consigue acelerar éste, en un ordenador portátil Intel i7, en un factor de 5 de forma general y en 7 veces en problemas de varios cientos de atributos. Teniendo en cuenta que en el tiempo total se contabiliza el tiempo de carga de disco del problema, lo que en dominios pequeños supone un porcentaje nada despreciable del tiempo total de proceso, vemos que con poco esfuerzo el uso de esta técnica es muy rentable computacionalmente.

2. Bajo uso de condicionales. Muchas otras técnicas agrupan patrones basándose en la distancia entre ellos. Calcular las distancias entre un patrón y todos los demás, requiere de un algoritmo de orden $O(N^2)$. Frente a ésto, el método de la rejilla es capaz de agrupar los patrones usando un algoritmo de orden N y, lo que también es muy importante, no requiere de condicionales. El uso de rejillas

[1] Un thread, hilo o hebra, tal como se conocen es un subproceso o proceso ligero que es la unidad mínima de trabajo que puede ser planificado por un sistema operativo. Al contrario que los procesos, los hilos comparten memoria y otros recursos entre todos los que componen el proceso principal, lo que aligera y acelera su uso. Gracias a esta característica son la técnica utilizada para sacar provecho a las capacidades multicore e hiperthread de las CPU modernas.

[2] Un procesador Intel i7 dispone de 4 cores y de hiperthreading (2 threads corriendo en paralelo en cada core). Así, para aprovechar al máximo esta CPU podemos hacer uso de $4 \times 2 = 8$ hilos.

posibilita agrupar N patrones, en una rejilla de un número arbitrario de dimensiones, simplemente realizando operaciones matemáticas sencillas, con números enteros y sin uso de condicionales. Para entender la importancia de este asunto, debemos recordar que uno de los avances más importantes en el rendimiento de las CPU modernas fue el desarrollo del pipeline o segmentación [1] de la CPU. En computación, se llama pipeline a una serie de elementos de procesamiento de datos ordenados, de tal modo, que la salida de cada uno es la entrada del siguiente, como una cadena de montaje, orientada al procesamiento de datos e instrucciones. Este importante elemento consigue que si una CPU tarda 30 ciclos de reloj en ejecutar una instrucción, con un pipeline suficientemente largo, podría ejecutar hasta 30 instrucciones cada 30 ciclos, lo que nos da una media de 1 instrucción/ciclo de reloj. Este elemento arquitectónico ha conseguido así mejorar espectacularmente el rendimiento de las CPU modernas pero, tiene un enemigo, que son las instrucciones que rompen el orden en que se ejecutan las instrucciones. Las instrucciones condicionales cambian el flujo de instrucciones ya se cumpla la condición o se haga falsa. Para paliar en lo posible esta circunstancia, las CPU modernas cuentan con una compleja circuitería que tiene como cometido predecir la rama de salto más probable. Si acierta la *pipeline* funciona perfectamente, si falla, decimos que se ha producido un fallo de predicción y, entonces, el rendimiento cae, ya que hay que vaciar la *pipeline* desde la instrucción de bifurcación en adelante. Cada condicional, especialmente si la probabilidad de tomar cada rama es equiprobable, es altamente perniciosa para el rendimiento total. Metafóricamente hablando, podemos comparar el ejecutar un algoritmo con condicionales equiprobables con la tarea de esprintar en una pequeña habitación, donde la cercanía a las paredes impide alcanzar velocidad.

Para ilustrar la importancia de este asunto y como ciertas estrategias utilizadas en la rejilla mejoran el rendimiento, se va a describir aquí un pequeño fragmento de código implementado de dos formas diferentes y se va comparar su rendimiento. En ambos casos, se presupone que el vector T contiene 0 ó 1 y que nuestro objetivo es computar cúantos 0 y cúantos 1 hay en total.

`CODIGO 1 (Con condicionales)`

[1] `http://es.wikipedia.org/wiki/Segmentacion_(informatica)`

```
long s=0;
for(it=0;it<N;it++) if (t[it]) s++;
printf("Unos:%u Ceros:%u\n",s,N-s);
```

```
CODIGO 2 (Sin condicionales-Utilizado en la rejilla)

long s[2]={0,0};
for(it=0;it<N;it++) s[t[it]]++;
printf("Unos:%u Ceros:%u\n",s[1],s[0]);
```

El *codigo*2, que no usa condicionales, se ejecuta 4 veces más rápido que el *codigo*1.

Este tipo de técnicas son usadas, profusamente, en la implementación hecha en la rejilla.

3. Localidad de accesos. La gran diferencia de velocidad entre la CPU moderna y el sistema de memoria se palía, en lo posible, haciendo uso de la jerarquía de memoria caché. La implementación de LIA puede realizarse de forma secuencial. De esta manera, la jerarquía de caché trabaja prácticamente de forma óptima lo que contribuye a un alto rendimiento y, a facilitar la implementación del punto siguiente de esta enumeración.

4. SIMD. Desde hace algo más de una década las CPU domésticas disponen de un juego de instrucciones con capacidades SIMD (una instrucción multiples datos). Esta técnica permite ejecutar, en paralelo, una instrucción sobre un vector de datos. Con el tiempo, el juego de instrucciones se ha ido ampliando y se dispone de varias familias de instrucciones: SIMD, SSE3, SSE3, SSE4, AVX, etc. Dado que LIA puede implementarse con sencillas operaciones aritméticas, suma y multiplicación, y el acceso a los datos en las rejillas puede organizarse para accesos secuenciales, la implementación de todos los pasos del algoritmo puede realizarse ventajosamente con instrucciones SIMD. En este trabajo de tesis y en lo que respecta a LIA, se ha preferido no complicar en exceso el código, para no caer

en sobreoptimización. No obstante, sí se ha experimentado con SSE3, codificando la fase de creación de una rejilla (código responsable de a partir de los datos del problema, rellenar una rejilla, para una combinación dada de atributos). Se ha conseguido una aceleración del código (para este paso concreto) de un 250 %. Previsiblemente, no existen obstáculos importantes para paralelizar de esta forma todo el algoritmo.

La implementación actual de LIA es capaz de procesar, aproximadamente, 400.000 rejillas por segundo, en un ordenador portátil de gama media.

3.3.4. Aplicabilidad de LIA

El parámetro *depth*, profundidad máxima a explorar, aporta cierto grado de flexibilidad a la técnica: si se requiere una respuesta muy rápida o tratar problemas con un número enorme de atributos, puede utilizarse $depth = 1$. Si se necesita la mejor precisión posible se utilizará $depth = 3$. $depth = 2$ será una configuración de compromiso, razonablemente rápida y precisa.

Con el fin de dar una indicación o referencia sobre el número máximo de atributos a tratar en cada profundidad, podemos guiarnos por la Tabla 3.4. Ha sido calculada considerando un tiempo máximo de respuesta de 30 segundos, un *hardware* consistente en un ordenador portátil Intel I5 con 4GB de memoria y problemas con 50 000 patrones de entrenamiento.

Tabla 3.4: Aplicabilidad de LIA

depth	Número máximo de atributos
1	4.500.000
2	3.000
3 optimizada	2.000
3	200

3.4. LIA2: Implementación de LIA con búsqueda *greedy*

LIA2, la variante *greedy* de LIA, se implementa como un algoritmo que va refinando progresivamente soluciones, incorporando a éstas, una regla en cada paso, hasta alcanzar la máxima profundidad de exploración deseada.

Llamamos aquí *solucion* a un conjunto de reglas unidas por claúsulas conjuntivas($\wedge$). Cada regla se compone de un atributo y un rango de valores para las cuales la regla se hace cierta. De esta forma, cada *solucion* delimita una región *n-dimensional*. Más formalmente, podemos definir *solucion* con la ecuación mostrada en la Ecuación 3.14.

$$solucion \equiv regla_1 \wedge regla_2 \wedge ... \wedge regla_n \tag{3.14}$$

donde cada $regla_i$, sigue la fórmula:

$$regla_i \equiv min \leq ATR_k \geq max \tag{3.15}$$

De esta forma, una *solucion* delimita una región del espacio, definida por rangos de valores de uno ó más atributos. A lo largo del texto, se usará en ocasiones la notación $1D$ para hacer alusión a soluciones de una sola regla y, por tanto, un solo atributo. Igualmente, se hablará de $2D$ y $3D$ para denotar soluciones que hagan uso de 2 y 3 reglas, respectivamente. Esta notación viene sugerida por el objeto geométrico al que da lugar una regla de 1, 2 ó 3 atributos: segmento (1D), rectángulo (2D) y paralelepípedo (3D).

LIA2, primeramente, crea una lista de las mejores soluciones computadas analizando el problema en 1D. Una vez creada dicha lista, se recorre la misma y, por cada solución 1D, se añaden soluciones 2D resultantes de refinar la solución 1D, añadiendo una regla más. El proceso se repite para todas las soluciones 1D. Una vez, refinadas todas, el proceso se repite hasta una profundidad de exploración arbitrariamente grande (parámetro *depth*). En la Figura 3.9 se representa este proceso de refinado incremental. Al final del proceso, la lista de soluciones resultante contendrá las mejores *soluciones*. El proceso se repite para diferentes valores de $confianza$. A partir del análisis de los valores de evaluación de cada solución se computa entonces una lista de atributos, ordenados de mayor a menor utilidad/relevancia para la resolución del problema.

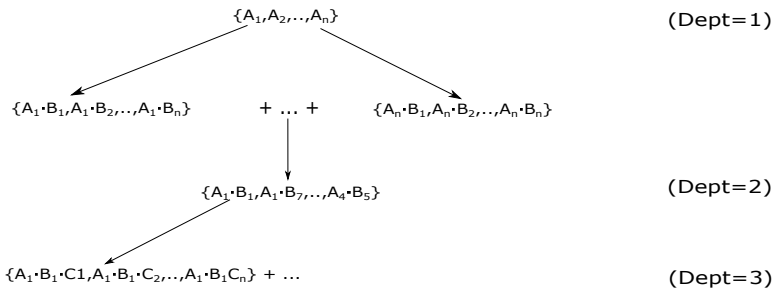

Figura 3.9: Refinado incremental de soluciones en LIA2

.

3.4.1. Algoritmo

LIA2 se describe en el Algoritmo 9. A lo largo de la explicación, se usará reiteradamente el concepto *solucion*. Se define aquí *solucion* como un objeto o estructura de datos que contendrá, además de las *reglas*, los campos: *valorEval* y δ*densidad*. *valorEval* almacenará el resultado de la evaluación. El campo δ*densidad* almacenará la *delta* calculada para dicha solución.

LIA2 calcula, para cada valor de $confianza$, la lista de mejores soluciones aplicables al problema. Dicha lista de soluciones, se obtiene haciendo uso de la función **MejoresSoluciones**, Algoritmo 11. Esta función va refinando progresivamente soluciones parciales, hasta que éstas alcanzan un máximo de *maxDepth* reglas. En la Sección 3.4.2 se puede consultar una descripción detallada de esta función. Como función de evaluación del celdas se utiliza la función $\mathbf{F_2}$ (ver Alg. 5). Dicha función se basa en valorar la ganancia de entropía, *Information Gain* (ver Ec. 3.11).

A continuación, se calcula con la función **ComputaRelevancia**, Algoritmo 10, la información correspondiente a cada atributo. Para ello, se asigna a cada atributo el mayor valor de evaluación, *valorEval*, de entre todas las reglas donde participe éste.

Seguidamente, se acumula dicha información en la matrix MI. MI tendrá tantas filas como posibles valores de $confianza$. Cada fila en MI almacena $nAtrs$ valores, correspondientes a la evaluación de cada atributo para un valor de $confianza$ dado.

Por último, la relevancia final de cada atributo se obtiene computando el sumatorio de las filas normalizadas de dicha matriz, utilizando la función **SumatorioFilasNormalizadas**, Algoritmo 8. VR contendrá entonces un vector de tantas componentes como atributos tenga nuestro problema, y cada elemento tendrá un valor que represen-

ta la relevancia de cada atributo. Si ordenamos estos por dichos valores obtendremos la lista de atributos más relevantes.

Algorithm 9 Algorithm LIA2

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo. Por defecto: 8.

depth: profundidad, número máximo de reglas de cada solución.

CP_{min}, CP_{max} valores de confianza mínimo y máximo.

CP_{step} tamaño del paso.

Output: VR: vector con $nAtrs$ elementos. El elemento $VR[i]$ contiene la relevancia del atributo i.

1: $C \leftarrow 0$
2: $ne \leftarrow Max(4, \sqrt{nAtrs})$
3: **for** *cp in* $[CP_{min}...CP_{max}$, *step* $CP_{step}]$ **do**
4: $\quad soluciones \leftarrow MejoresSoluciones(X, Y, P, 1 - 10^{-cp}, depth, nAtrs, ne, F_2)$
5: $\quad MI[C] \leftarrow ComputaRelevancia(soluciones, nAtrs)$
6: $\quad C \leftarrow C + 1$
7: **end for**
8: $VR \leftarrow SumatorioFilasNormalizadas(MI, C, nAtrs)$
9: **return** VR

En el resto de capítulos de esta tesis, se ha utilizado y experimentado con LIA2 tal como está definida en este capítulo. No obstante, se apunta como posibilidad añadir, en el paso número 27 de la definición de la función **MejoresSoluciones**, un paso opcional llamado Reevaluar. El objetivo de dicho paso opcional sería reevaluar cada una de las soluciones contenidas en listaSoluciones, con una segunda función de evaluación de carácter finalista. $\mathbf{F_2}$ resulta de gran utilidad como función de evaluación generalista y, particularmente, como función de búsqueda de soluciones parciales, soluciones que pueden ser refinadas en pasos incrementales. Sin embargo, resulta concebible que la reevaluación con una función, $\mathbf{F_R}$, que tenga caracter finalista (esto es, una función que evalúe exactamente la medida que queremos optimizar) proporcione un ajuste fino y mejores resultados. Así, $\mathbf{F_R}$ sería cualquier función de evaluación que proceda primeramente calculando el número de patrones fiables. Para ello, hará uso de la

Algorithm 10 FUNCTION ComputaRelevancia

Input:

listaSoluciones: lista de objetos de tipo Solución. *nAtrs*: número de atributos.

Output: VR: vector con *nAtrs* elementos. El elemento $VR[i]$ contiene la relevancia del atributo i.

```
1: VR ← new Vector(nAtrs,0.0)
2: for solucion in [listaSoluciones] do
3:     for atr in [solucion.getAtrs()] do
4:         v ← solucion.valorEval
5:         if VR[atr] < v then
6:             VR[atr] ← v
7:         end if
8:     end for
9: end for
10: return VR
```

$\delta Densidad$ ya calculada para la *solucion* en los pasos previos del algoritmo (ver Función **grid.mejorSolucion** en 3.4.2). Una vez calculados los patrones fiables, se evaluarían con otra función que devolvería un valor numérico que pasaría a ser, a partir de ese momento, el valor de *valorEval* almacenado en la *solucion*. Ejemplos de funciones que pueden ser de utilidad como $\mathbf{F_R}$: *precision* (ver Ec. 2.3), *recall* (ver Ec. 2.4), *F-Score* (ver Ec. 3.16) y PSR (ver Ec. 3.17). Esta última sería una función hecha a medida, que combina *precison* y *recall*, de forma que priorice *precision* pero valore en cierta medida también *recall*.

$$FScore = \frac{2 \times precision() \times recall()}{precision() + recall()} \tag{3.16}$$

$$Psr = precision() \times \frac{1}{1 + e^{(-(recall()-0{,}125)*25)}} \tag{3.17}$$

3.4.2. Función MejoresSoluciones

Dada la importancia y complejidad de esta función y la reutilización que se hace de la misma por varias de las técnicas presentadas, se presenta aquí una explicación detallada de su funcionamiento. Como puede observarse, la función primeramente computa

soluciones en una dimensión (una única regla). Después, se van refinando progresivamente éstas, añadiendo reglas con atributos de uno en uno. Ésto es, el algoritmo hace una búsqueda de soluciones en anchura. Éstas son generadas refinando cada *solucion* 1D añadiendo una regla. Por cada *solucion* se almacena en *listaSoluciones* un máximo de *nExpand* soluciones ampliadas. El tamaño máximo permitido de *listaSoluciones* se va variando según avanzamos en profundidad, pero siempre limitado a un número de soluciones menores a las resultantes de refinar todas las *soluciones* presentes en *listaSoluciones nExpand* veces. De esta forma, se hace competir a las *soluciones* entre sí, de forma que solo las mejores van permaneciendo. Se intenta con esto mantener cierto equilibrio entre mantener las mejores *soluciones* y, a la vez, retener cierta diversidad. Gracias a este procedimiento, el algoritmo es capaz de sortear mínimos locales.

A continuación se explican algunas de las funciones auxiliares utilizadas por la función $MejoresSoluciones$:

grid.mejorSolucion($confianza, densidad, funcionEval$**)**: computa para cada celda de un *grid* el valor de la función $funcionEval$ ($F2$ en LIA2). La función devuelve la *solucion* que define la celda que hace máxima la función de evaluación para valores de $confianza$ y *densidad* dados. El objeto *solucion*, resultado de la función, contiene además de las reglas dos campos: $valorEval$ y $\delta densidad$. El campo $valorEval$ almacenará el resultado de la evaluación de la hipercelda, calculado con la función $funcionEval$. Y, el campo $\delta densidad$ almacenará la *delta* calculada para dicha *solucion*.

lista.insert($solucion, max$**)**: la función *insert*, inserta una *solucion* en una lista de soluciones, ordenada por un valor numérico correspondiente al valor del campo $valorEval$ de cada *solucion*. La función garantiza que en todo momento *lista* solo contendrá un máximo de max elementos. En caso de que el número de soluciones supere el límite max la función garantiza descartar siempre las peores soluciones, siendo éstas las que tengan $valorEval$ inferiores. **lista.insertLista(**$lista2, max$**)**: funciona de manera similar a $lista.insert(solucion, max)$ pero iterando sobre cada elemento contenido en $lista2$.

grid.rellena($problema, atributo$**)**: computa una rejilla, *grid*, proyectando los valores del *atributo* para todos los patrones de un problema, sobre una rejilla unidimensional. En cada celda de la rejilla, se almacenan el número de patrones de clase C_1 y C_0 proyectados sobre la misma.

Algorithm 11 FUNCTION **MejoresSoluciones**

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo.

$confianza$: valor de confianza.

$maxDepth$: profundidad, número máximo de reglas de cada solución.

$maxSol1D$: máximo número de soluciones de primer nivel.

$nExpand$: coeficiente de expansión de cada solución.

F: función utilizada para evaluar cada solución.

Output: La función devuelve la lista de mejores soluciones.

1: $listaSoluciones \leftarrow new\ ListaSoluciones()$
2: $grid \leftarrow newGrid(P, 1)$
3: $densidad \leftarrow GetDensidad(Y)$
4: **for** atr in [1..nAtrs] **do**
5: grid.rellena(X,Y,atr)
6: $solucion \leftarrow grid.mejorSolucion(confianza, densidad, F)$
7: $listaSoluciones.insert(solucion, maxSol1D)$
8: **end for**
9: **for** depth in [2..maxDepth] **do**
10: $tamMax \leftarrow |listaSoluciones| \times nExpand \div 2$
11: **for** solucion in [listaSoluciones] **do**
12: **if** solucion.getDimension() == (depth-1) **then**
13: $lista \leftarrow new\ ListaSoluciones()$
14: $(XF, YF) \leftarrow Seleccionar(X, Y, solucion)$
15: $densitySol \leftarrow GetDensidad(YF)$
16: **for** atr in [1..numAtrs] **do**
17: **if** not solucion.contieneAtributo(atr) **then**
18: $grid.rellena(XF, YF, atr)$
19: $solucion \leftarrow grid.mejorSolucion(confianza, densitySol, F)$
20: $lista.insert(solucion, nExpand)$
21: **end if**
22: **end for**
23: $listaSoluciones.insertLista(lista, tamMax)$
24: **end if**
25: **end for**
26: **end for**
27: **(Opcional) Reevaluar(listaSoluciones,F_R)**
28: **return** $listaSoluciones$

3.5. LIA3: Implementación de LIA como árbol de decisión

La técnica LIA3 se implementa como un árbol de decisión binario donde cada nodo contiene una regla definida de forma similar a la Ecuación 3.15. Como función utilizada como decisor, para evaluar si es conveniente expandir un nodo, se utiliza la función $\mathbf{F_2}$ (ver Alg. 5). Una de las novedades que presenta la construcción de dicho árbol, consiste en la forma de decidir si expandir o no un nodo. En las secciones siguientes se detallará cómo LIA3 hace uso de un procedimiento de exploración adelantada para dotar de mayor robusted al proceso de búsqueda de regla de expansión. Con ello, LIA3 es capaz de superar adecuadamente mínimos locales. Una vez generado el árbol correspondiente, podemos obtener la lista de atributos, ordenados por relevancia, podando el árbol de forma tal que vayamos retirando los nodos en orden inverso al valor de ganancia de entropía global que aportan al árbol. Así, las reglas usadas y, por tanto, los atributos usados en éstas, para los primeros nodos en ser retirados serían los atributos menos relevantes y los últimos los más relevantes. Dado que un atributo puede aparecer en distintos nodos del árbol, su relevancia vendrá dada por el orden en que sea retirado el último nodo que haga uso del mismo.

De forma similar a LIA2, el algoritmo itera para distintos valores de $confianza$. La lista final de atributos relevantes se obtiene combinando las listas obtenidas para cada valor de $confianza$ evaluado.

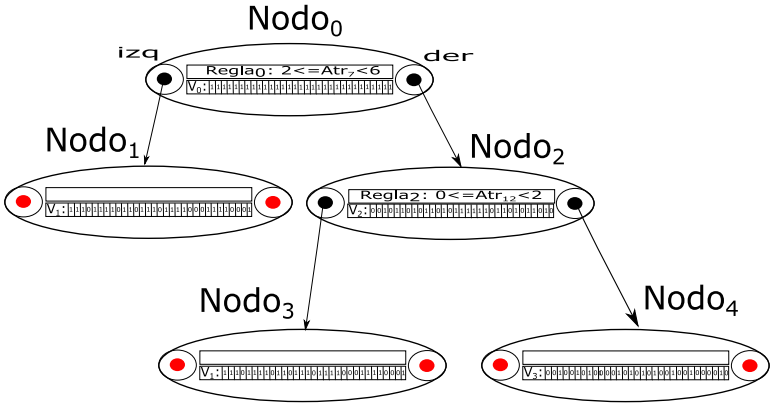

Figura 3.10: Árbol de decisión LIA3

.

3.5.1. Algoritmo

La primera fase del algoritmo LIA3 tiene como objetivo crear un árbol binario, como el presentado en la Figura 3.10. A lo largo del siguiente texto, se va a usar esta nomenclatura: llamamos *hoja* o *nodo terminal* a aquellos nodos que no tienen descendientes ni, por tanto, regla de particionamiento (en el ejemplo mostrado en la Figura 3.10, $Nodo_1, Nodo_3, Nodo_4$ serían *nodos terminales*). Por último, *Nodo Split Terminal* será aquel cuyos dos descendientes sean nodos terminales. En el ejemplo, $Nodo_2$ sería un *nodo split terminal*. Cada nodo del árbol viene representado por una estructura que contendrá los siguientes campos:

- Una regla de decisión, *regla*, con formato similar al descrito en la Ecuación 3.15. Los patrones que hagan positiva dicha regla, pasarán a su descendiente derecho y los que no a la izquierda.
- *der* e *izq*, son apuntadores a los nodos descendientes.
- Un vector de números binarios, llamado V, que representará los patrones responsabilidad de nicho nodo. '1' indicará que el patrón está seleccionado y '0' que no lo está. El primer nodo del árbol contendrá un vector V con todos los patrones seleccionados, todos '1'. A partir de él, todos los descendientes irán seleccionando patrones haciendo uso de las reglas de decisión.

A lo largo del algoritmo se hace uso de algunas funciones de uso habitual en la disciplina aplicables a Vectores y/o Listas. Siendo éstas de uso común se omite aquí una explicación detallada de las mismas. También, se utilizan algunas funciones auxiliares aplicables a objetos de tipo *Solucion* (ver Ec. 3.14) y, otras dedicadas a seleccionar patrones por diferentes criterios.

Funciones aplicables a *Solucion*:

- Solucion.firstRule(): Devuelve la primera regla que compone la solución.

Otras funciones auxiliares:

- Seleccionar(X,Y, Regla r): Devuelve una pareja de subconjuntos de X e Y, donde solo aparecen los patrones que corresponden a filas donde se hace cierta la regla r.

- Seleccionar(X,Y, Vector v): Devuelve una pareja de subconjuntos de X e Y, donde solo aparecen los patrones que corresponden a filas donde $v[fila] = 1$.
- GetVectorSel(X,Vector v,Regla r): Devuelve un vector binario, donde cada elemento i toma el valor 1 si $v[i] = 1$ y el X[i] hace cierta la regla r.
- GetCell(Y,Vector v): Devuelve una celda, donde C1 es el número de patrones de clase C1 y C0 el número de patrones de clase C0, presentes en el vector Y y, tal que, $v[i] = 1$.

El algoritmo principal que implementa LIA3, se describe en el Algoritmo 13.

LIA3 ejecuta un bucle donde va entrenando, para cada valor de confianza, un árbol de decisión. Una vez obtenido éste, se calcula la lista de atributos relevantes, en base a la ganancia de entropía que aporta cada atributo al árbol completo. El orden de los atributos en la lista de relevancia se almacena en una matriz llamada MI. MI tendrá tantas filas como posibles valores de $confianza$. Cada fila en MI almacena $nAtrs$ valores, correspondientes al orden de cada atributo para un valor de confianza dado. Siendo el atributo más relevante, para un valor de confianza dado, el atributo con número de orden $nAtrs$ y el menos relevante con valor 1.

A continuación, la lista final de atributos se obtiene computando el sumatorio de las filas normalizadas de dicha matriz, haciendo uso de la función **SumatorioFilasNormalizadas** (ver Alg. 8). VR contendrá entonces un vector de tantas componenes como atributos tenga nuestro problema, y cada elemento tendrá un valor que representa la relevancia de cada atributo. Si ordenamos estos por dichos valores obtendremos la lista de atributos más relevantes.

Por su parte, la función **Fit** es la encargada de entrenar un árbol de decisión con los patrones contenidos en X e Y, para un valor concreto de $confianza$.

La función **Expandir** es la encargada de decidir si se deben particionar los patrones responsabilidad del nodo, aplicando una regla de decisión o si el nodo se convierte en un nodo terminal. Si es posible encontrar algún criterio de partición (solución), que aporte una mejora en el valor de la función de evaluación, el nodo se divide en dos nodos hijos. En ese caso, el nodo apuntado *der* contendrá los patrones de datos seleccionados por la nueva regla y el apuntador izq contendrá el resto de patrones.

A continuación, la **GetAtrsMoreRelevant** es la encargada de obtener una lista de atributos, ordenados de mayor a menor relevancia, a partir de un árbol de decisión.

Para ello, va eliminando nodos terminales, en order inverso a la ganancia de información que aporta cada nodo.

La función **GetPeorNodoSplitTerminal** se encarga de seleccionar, recursivamente, el nodo que aporta menos ganancia de información globalmente.

Para calcular la ganancia de información, que proporciona cada nodo, se hace uso de la función **Gain**. El valor numérico devuelto cuantifica la ganancia global de información o reducción de entropía. La función hace uso de la función $\mathbf{F_2}$ (ver Alg. 5).

Algorithm 12 LIA3

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo. Por defecto: 8.

$depth$: profundidad de exploración.

CP_{min}, CP_{max}: valores de confianza mínimo y máximo.

CP_{step}: tamaño del paso.

$maxDepthTree$: máxima profundidad del árbol de decisión.

Output: VR: vector con $nAtrs$ elementos. El elemento $VR[i]$ contiene la relevancia del atributo i.

1: $C \leftarrow 0$
2: **for** *cp in* $[CP_{min}..CP_{max},$ *step* $CP_{step}]$ **do**
3: $\quad arbol \leftarrow Fit(X, Y, P, 1 - 10^{-cp}, depth, maxDepthTree)$
4: $\quad relevantes \leftarrow GetAtrsMoreRelevant(arbol)$
5: $\quad orden \leftarrow nAtrs$
6: $\quad$ **for** *atr in relevantes* **do**
7: $\quad\quad MI[C, atr] \leftarrow orden$
8: $\quad\quad orden \leftarrow orden - 1$
9: $\quad$ **end for**
10: $\quad C \leftarrow C + 1$
11: **end for**
12: $VR \leftarrow SumatorioFilasNormalizadas(MI, nAtrs, C)$
13: **return** VR

Algorithm 13 LIA3 (continuado)

14: **function Fit**

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo.

$confianza$: valor de confianza.

$depth$: profundidad de exploración.

$maxDepthTree$: máxima profundidad del árbol de decisión.

Output: La función retorna un apuntador al nodo raíz del árbol.

15: $nodoRoot \leftarrow new\ Nodo()$
16: $nodoRoot.depth \leftarrow 1$
17: $nodoRoot.v \leftarrow new\ Vector()$
18: $nodoRoot.v.setAll(1)$
19: $pendientes \leftarrow new\ Lista()$
20: $pendientes.append(nodoRoot)$
21: **while** $\neg pendientes.vacio()$ **do**
22: $nodo \leftarrow pendientes.pop_back()$
23: **if** $nodo.depth < maxDepthTree$ **then**
24: **if** $Expandir(nodo, X, Y, P, confianza, depth)$ **then**
25: $pendientes.append(nodo.der)$
26: $pendientes.append(nodo.izq)$
27: **end if**
28: **end if**
29: **end while**
30: **return** $nodoRoot$
31: **end function**

Algorithm 14 LIA3 (continuado)

32: **function Expandir**

Input:

nodo: apuntador a un nodo del árbol.

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

P: número de intervalos de discretización de cada atributo.

$confianza$: valor de confianza.

$depth$: profundidad de exploración.

Output: La función devuelve TRUE si el nodo ha sido expansionado y, FALSE en caso contrario.

33: $ne \leftarrow Max(4, \sqrt{nAtrs})$
34: **if** $\neg nodo.v.isUniforme()$ **then**
35: $(XF, YF) \leftarrow Seleccionar(X, Y, nodo.v)$
36: $sols \leftarrow MejoresSoluciones(XF, YF, P, confianza, depth, nSols1dMax, ne)$
37: **if** $|sols| > 0$ **then**
38: $solucion \leftarrow mejorSolucion(sols)$
39: $rule \leftarrow solucion.firstRule()$
40: $nodo.rule \leftarrow rule$
41: $nodo.izq \leftarrow new\ Nodo()$
42: $nodo.izq.depth \leftarrow nodo.depth + 1$
43: $nodo.der \leftarrow new\ Nodo()$
44: $nodo.der.depth \leftarrow nodo.depth + 1$
45: $nodo.der.v \leftarrow GetVectorSel(X, nodo.v, rule)$
46: $nodo.izq.v \leftarrow nodo.der.v.not()$
47: **return** $TRUE$
48: **end if**
49: **end if**
50: **return** $FALSE$
51: **end function**

Algorithm 15 LIA3 (continuado)

52: **function GetAtrsMoreRelevant**

Input:

arbol: Será un apuntador al primer nodo, o nodo raíz del árbol de decisión.

Output: La función devuelve una lista de atributos, ordenados por grado de relevancia, de mayor a menor.

```
53:     atributos ← new Lista()
54:     while ¬arbol.vacio() do
55:         peor ← GetPeorNodoSplitTerminal(arbol)
56:         atr ← peor.getAtributo()
57:         if atributos.contiene(atr) then
58:             atributos.eliminar(atr)
59:         end if
60:         atributos.append(atr)
61:         eliminarNodoSplit(peor)
62:     end while
63:     atributos.invertir()
64:     return atributos
65: end function
```

Algorithm 16 LIA3 (continuado)

66: **function GetPeorNodoSplitTerminal**

Input:

nodo: apuntador a un nudo del árbol.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

$confianza$: valor de confianza.

Output: Devuelve el NodoSplit que proporciona la menor ganancia de información, para *nodo* y todos sus descendientes. Si no es posible calcularlo devuelve NULL.

67: **if** $nodo.isTerminal()$ **then**
68: **return** $NULL$
69: **end if**
70: **if** $nodo.izq.isTerminal() \wedge nodo.der.isTerminal()$ **then**
71: **return** $nodo$
72: **end if**
73: $peorIzq \leftarrow GetPeorNodoSplitTerminal(nodo.izq, Y, confianza)$
74: $peorDer \leftarrow GetPeorNodoSplitTerminal(nodo.der, Y, confianza)$
75: **if** $peorIzq = NULL$ **then**
76: **return** $peorDer$
77: **end if**
78: **if** $peorDer = NULL$ **then**
79: **return** $peorIzq$
80: **end if**
81: **if** $(Gain(peorIzq, confianza, Y) < Gain(peorDer, confianza, Y))$ **then**
82: **return** $peorIzq$
83: **else**
84: **return** $peorDer$
85: **end if**
86: **end function**

87: **function Gain**

Input:

nodo: es un apuntador al nodo terminal a evaluar.

Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.

$confianza$: valor de confianza.

Output: La función devuelve un número que representa la ganancia de información.

88: $espacio \leftarrow getCell(Y, nodo.v)$
89: $sel \leftarrow getCell(Y, nodo.v.der)$
90: **return** $F_3(sel, confianza, espacio)$
91: **end function**

3.6. Comparativa entre LIA, LIA2 y LIA3

Cada una de las tres variantes de LIA aporta unas ventajas o puntos fuertes y unas desventajas o inconvenientes para su uso como técnicas de selección de atributos. A continuación se enumeran esquemáticamente las principales características e inconvenientes que presenta cada una.

- LIA:
  - Exploración exhaustiva en dimensión 1D, 2D y/o 3D. Esto implica:
    - Evalúa todas las combinaciones de atributos.
    - Puede encontrar información sin necesidad de que por monoticidad se encuentre información en dimensiones bajas.
    - Es la versión más sensible (es capaz de encontrar más zonas de alta densidad) y precisa (es capaz de definir de forma más precisa las zonas de interés). Ambas características facilitan encontrar los atributos más relevantes, especialmente, en problemas de Tipo-I.
  - Inconvenientes:
    - Es la más lenta de todas (cuando exploramos en 2D y 3D). Aplicable, en la práctica, hasta 200 atributos en 3D, 2.000 atributos en 2D y en 3D optimizada.
    - No es robusta a atributos redundantes.
    - No garantiza encontrar la combinación de atributos que globalmente optimice una medida dada.
- LIA2:
  - Búsqueda *greedy* desde dimensión 1 hasta dimensión *depth*. Va refinando las soluciones encontradas, paso a paso. Puede explorar hasta profundidades arbitrariamente altas.
  - Puede ser rápida (expandiendo pocos nodos) o precisa (expandiendo muchos nodos en cada nivel).
  - Puede optimizar cualquier función de evaluación local.

- Aplicable a problemas con un número muy alto de atributos (alta dimensionalidad).
- Inconvenientes:
  - Si el problema presenta poca monoticidad puede ser menos precisa que LIA.
  - No es robusta a atributos redundantes.
  - No garantiza encontrar la combinación de atributos que globalmente optimicen una medida dada.

- LIA3:
  - Búsqueda *greedy* desde dimensión 1 hasta dimensión $maxDepthTree$ usando una estructura de árbol que va creciendo nivel a nivel.
  - Puede explorar hasta dimensiones arbitrariamente altas.
  - Realiza una exploración de soluciones, adelantada *depth* niveles, capaz de superar mímimos locales.
  - Puede usar una función de optimización local (evalúa una hoja) y otra global (evalúa todo el árbol).
  - Aplicable a problemas con un número muy alto de atributos (alta dimensionalidad).
  - Es robusta a atributos redundantes.
  - En principio, podría generar un árbol razonablemente bueno para optimizar globalmente cualquier medida. El árbol resultante, podría ser, en su caso, utilizable como algoritmo de clasificación.
  - Inconvenientes:
    - Si el problema presenta poca monoticidad puede ser menos precisa que LIA.
    - Si las primeras decisiones en el árbol son incorrectas el resultado final puede ser sub-óptimo.

En la Tabla 3.5 se resumen las características principales de cada variante de LIA.

Tabla 3.5: Comparativa versiones LIA

Característica	LIA	LIA2	LIA3
Búsqueda exhabustiva	Sí (hasta 3 dimensiones)	No	No
Máximo rendimiento en problemas Tipo-I	Sí	No	No
Requiere monoticidad hasta 1D	No	Sí	Sí
Muy rápida	No (Sí para depth=1)	Sí	Sí
Aplicable a problemas de muy alta dimensionalidad	Solo si depth=1	Sí	Sí
Robusta a atributos redundantes	No	No	Sí
Optimiza solución global	No	No	Sí

3.7. LIAR: LIA para problemas de regresión

Este desarrollo intenta sacar partido a las técnicas de selección de atributos presentadas con anterioridad: LIA, LIA2 y LIA3, adaptando los problemas de regresión de forma tal que sean abordables por éstas. Para ello, el problema de regresión es convertido en *intervalosOut* problemas binarios, discretizando para ello la salida, en *intervalosOut* intervalos de valores discretos. Por defecto, el parámetro *intervalosOut* toma el valor 10. Una vez calculada la lista de atributos más relevantes para cada uno de los intervalos, una última fase de la técnica combina dichas listas generando una única lista de atributos ordenada por relevancia.

3.7.1. Algoritmo

En el Algoritmo 26 se describe la técnica LIAR. Ésta consta de las tres fases que se describen a continuación:

1. Discretización del valor de salida. La función **DiscretizaClase** discretiza el valor de salida utilizando intervalos de ancho constante. En caso de que el número de patrones en cada intervalo no sea uniforme se discretizará el valor de salida de forma que el número de patrones de cada clase sea igual para todos los intervalos. **DiscretizaClase** hace uso de la función **IsUniforme** (ver Alg. 3).

2. Selección de atributos para cada intervalo por separado. Una vez discretizado el problema y convertido a *intervalosOut* problemas binarios, se utiliza LIA para obtener la lista de atributos relevantes para cada valor discreto.

3. Combinación de los atributos obtenidos para cada intervalo en una única lista de atributos. Para realizar la combinación de las listas de atributos relevantes se procede de la siguiente manera:

   3.1. Se selecciona el primer atributo, el atributo más relevante, de cada lista.

   3.2. Se ordena la lista de atributos obtenidos, atendiendo a la relevancia global o suma de relevancia de cada atributo para el total de intervalos.

   3.3. Los atributos de dicha lista, se añaden, en orden, a la lista global de atributos relevantes, siempre que no estuvieran ya en dicha lista.

   3.4. Se itera el procedimiento para el segundo nivel de relevancia hasta el último nivel o hasta que la lista de resultados contenga ya el número suficiente de atributos.

Algorithm 17 LIAR

Input:

X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.

Y: vector de $nPatrones$ elementos que contiene el valor de salida de cada patrón.

$nParticiones$: establece en cuantos intervalos se discretizará la clase de salida. Por defecto, tomará el valor de 10.

Output: *relevantes* es un vector que contendrá los atributos seleccionados ordenados de mayor a menor relevancia.

```
1: vectorDiscretizado ← DiscretizaClase(Y, nParticiones)
2: for a in [0,nAtrs-1] do
3:     vectorRelevanciaGlobal[a] ← 0
4: end for
5: for i in [0,nParticiones-1] do
6:     vectorAtributos_i = LIA(X, vectorDiscretizado = i)
7:     for a in [0,|vectorAtributos|] do
8:         relevancia ← nAtrs − a
9:         atr ← vectorAtributos[a]
10:        if vectorRelevanciaGlobal[atr] < relevancia then
11:            vectorRelevanciaGlobal[atr] ← relevancia
12:        end if
13:    end for
14: end for
15: for profundidad in [0,nAtrs-1] do
16:    for i in [0,nParticiones-1] do
17:        v[i] = vectorAtributos_i[profundidad]
18:    end for
19:    Sort(v,by vectorRelevanciaGlobal)
20:    for i in [0,nParticiones-1] do
21:        if (v[i] not in relevantes) then
22:            relevantes.append(v[i])
23:        end if
24:    end for
25: end for
26: return relevantes
```

Algorithm 17 LIAR (continuado)

27: **function DiscretizaClase**

Input:

Y: vector que contiene el valor de salida para cada patrón de datos.

$intervalosOut$: es un parámetro que establece en cuantos intervalos se discretizará la clase de salida. Por defecto, tomará el valor de 10.

Output: $vectorDiscretizado$ es un vector que contendrá los valores del atributo clase discretizados. Tomará valores $vectorDiscretizado_i \in [0, intervalosOut - 1]$.

28: $hist_1 \leftarrow histogramaWidthConstante(Y, intervalosOut)$
29: **if** $isUniforme(hist_1)$ **then**
30: $vectorDiscretizado = hist_1.transform(Y)$
31: **else**
32: $hist_2 \leftarrow histogramaHeightConstante(Y, intervalosOut)$
33: $vectorDiscretizado = hist_2.transform(Y)$
34: **end if**
35: **return** $vectorDiscretizado$
36: **end function**

3.8. CLIA: Clasificación por análisis local de la fiabilidad

Las técnicas de selección de atributos que se han desarrollado para esta tesis, LIA, LIA2, LIA3 y LIAR, pueden considerarse generalistas, pudiendo utilizarse en todo tipo de problemas.

Basada en conceptos muy similares, se ha desarrollado, también, una técnica de clasificación denominada Clasificación por Análisis Local de la Fiabilidad (CLIA). Dicha técnica se ha concebido con la única finalidad de abordar el objetivo I de esta tesis: *predicción en el Mercado de Valores*. Por lo que el diseño realizado está adaptado a dicho problema.

Específicamente, la nueva técnica se ha diseñado para atender los requisitos identificados en dicho problema (ver Sección 1.2). A modo de resumen, los requisitos para la nueva técnica se pueden enumerar como: robusta ante escasez de patrones, atributos irrelevantes y presencia de poca información *vs* ruido.

El objetivo del clasificador será encontrar la zona de máxima densidad de patrones de clase C_1. Dicho objetivo puede establecerse también como: encontrar la zona de máxima *precision* para C_1. En la siguiente sección se describirán las diferentes fases en que se estructura el algoritmo de clasificación propuesto.

3.8.1. Estructura general

La estructura de la nueva técnica sigue, así, las siguientes fases:

- Fase I: Selección de Atributos.
- Fase II: Clasificación.
  - Fase II.I: Fase exhaustiva combinatoria.
  - Fase II.II: Fase agregativa.
  - Fase II.III: Fase de consolidación.

Una vez entrenado, el clasificador dispondrá de una lista de soluciones parciales, donde cada *solucion*, S_i, resolverá parte del problema.

Cada *solucion* S_i estará definida según la Ecuación 3.18 y, estará formada por claúsulas unidas con disyunciones ($\wedge$).

$$solucion_i = (a \wedge b \wedge c \wedge ...) \tag{3.18}$$

donde cada claúsula, a,...,j, sigue la Ecuación 3.19.

$$a \equiv min \leq ATR_k \geq max \tag{3.19}$$

Siendo ATR_k el atributo que ocupa la posición k en el patrón de entrada a clasificar.

El modelo interno del clasificador, utilizado para clasificar cada patrón de datos, será así un conjunto de reglas, Ecuación 3.20, que seguirán la forma normal conjuntiva (FNC).

$$clasificador = S1 \vee S2 \vee S3... = (a \wedge b \wedge c \wedge ...) \vee (d) \vee (e \wedge f) \vee (g \wedge h \wedge ...) \vee ... \tag{3.20}$$

Todo patrón que haga cierta dicha fórmula producirá como salida del clasificador un 1. La salida será 0 en caso contrario.

Dado que la técnica se va a aplicar en ámbitos como el mercado de valores, donde, se piensa, existen pocos patrones informados, se crea el estado *Indeterminado*, con el que el clasificador expresa que no existe información utilizable en el problema para predecir la clase de salida. Así, un '1' indicaría la predicción de un patrón de clase C_1. Por contra, un '0' indicaría que no se dispone de información para realizar una predicción. Por lo tanto, un '0' no indica, necesariamente, patrón de clase C_0. La Tabla 3.6 resume el significado semántico de la salida del clasificador.

Salida del Clasificador	Predicción
1	Clase C_1
0	*Indeterminado*

Tabla 3.6: Significado semántico de la salida del clasificador

3.8.2. Principales parámetros de la técnica

- Parámetros para Fase I - Selección Atributos:
  - *pcSelAtrs*: número de atributos a seleccionar en la Fase-I. Puede ser expresado como número o como porcentaje sobre el total de atributos del problema. Valor por defecto: 5.
- Parámetros para Fase II - Clasificación:
  - *nExpand*: máximas expansiones en fase II.II, agregativa, por cada solución a refinar. Valor por defecto: 4.
  - *depth*: máxima profundidad a explorar en fase II.II. Valor por defecto: 2.
  - $confClasif$: valor del parámetro $confianza$ utilizado en la fase de clasificación. Valor por defecto: 5.

3.8.3. Fase I: Selección de Atributos

Dado que CLIA implementa una búsqueda de soluciones exhaustiva, es conveniente preceder los cálculos de una fase de selección de atributos, de forma que la fase combinatoria se ejecute solo sobre un conjunto de atributos relativamente pequeño. Es decir, se realiza la búsqueda exhaustiva solo sobre el conjunto de atributos que *a priori* tiene más posibilidades de contener a los mejores. Así, esta fase tiene como objetivo identificar, rápidamente, los atributos más relevantes para la clasificación de la clase minoritaria.

Podría utilizarse cualquier método de selección de atributos para completar esta primera fase. LIA parece ser una técnica competitiva para identificar atributos que aportan información fiable frente a un gran número de atributos irrelevantes y/o ruiodos. Por ello, para la implementación de esta fase I, CLIA implementa un algoritmo similar al descrito para LIA (ver 3.3), parametrizado con $mode = 1$.

Como resultado de esta primera fase, tendremos una lista de N atributos, ordenados por relevancia.

3.8.4. Fase II: Clasificación

El objetivo de esta fase es obtener una/s reglas lógicas que realicen eficazmente la clasificación de los patrones de datos, en una de las dos clases de salida del problema.

Para abordar este objetivo, la fase II se divide en tres subfases: Fase II.I Exhaustiva combinatoria, Fase II.II Agregativa y II.III Fase de consolidación.

En la fase exhaustiva combinatoria, se exploran todas las combinaciones de hasta tres atributos seleccionados en la fase I. De esta forma, la técnica enfoca su esfuerzo en los atributos donde *a priori* puede resultar más productiva una búsqueda exhaustiva. Por razones de eficacia (en tiempo de proceso), no es recomendable que el número de atributos con los que se trabaja en esta fase II.I sea demasiado elevado.

Una vez obtenida una lista de soluciones, utilizando la búsqueda combinatoria se intenta optimizar o refinar cada *solucion* añadiendo a ésta reglas formadas por cualquiera de los atributos del problema. En esta fase II.II, se hace uso de una estrategia *greedy* o incremental. De esta forma, se van añadiendo reglas a cada *solucion* en tanto en cuanto, la regla añadida suponga una mejora en cuanto a valor de evaluación de la *solucion*.

Por último, en la fase II.III, se eliminan, iterativamente, de la lista de soluciones las que no supongan una mejora de los resultados. El producto final de esta última fase, es una lista de soluciones que aplicadas conjuntamente proporcionan el mejor valor de evaluación de la medida a optimizar (en esta tesis la medida utilizada es *precision*).

3.8.5. Fase II.I: Fase exhaustiva combinatoria

En esta fase se exploran, de forma exhaustiva, todas las posibles combinaciones de 1, 2 y 3 atributos de entre los seleccionados en la fase previa. Para ello, se utiliza una rejilla como método de búsqueda de soluciones (ver Sección 3.1.5). Se computan todas las posibles hiperceldas que puedan inscribirse en cada rejilla. Se entiende por hipercelda (ver Sección 3.1.4), todo paralelepípedo, formado por la unión de celdillas individuales, de posición y dimensiones tales que puedan inscribirse dentro de la rejilla n-dimensional.

A continuación se muestran los pasos seguidos por el algoritmo, en esta fase de búsqueda exhaustiva:

1. Variable dim=1.

2. Para cada combinación de *dim* atributos de entre los seleccionados en la fase I, se construye una rejilla de 8^{dim} celdas y se computa el número de patrones de cada clase pertenecientes a cada celda. Para ello se divide el espacio en regiones cuadradas de 0.125 de lado, lo que divide el espacio de entrada en 8 regiones por cada dimensión.

3. Para cada rejilla se computan todas las posibles agrupaciones de celdas en hiperceldas.

4. Haciendo uso de la función de evaluación $\mathbf{F_3}$ (ver Alg. 18), se selecciona la hipercelda que presente el valor de evaluación más alto.

5. Se actualiza, si es necesario, la lista de soluciones con la *solucion* definida por la hipercelda.

6. Se repite el proceso para *dim*=2 y *dim*=3.

3.8.5.1. Lista de soluciones

El objetivo de este componente es ir almacenando las mejores soluciones encontradas. Dado que las soluciones almacenadas serán refinadas en fases posteriores del algoritmo, resulta conveniente intentar mantener cierta diversidad. La mejor solución final no tiene porque ser la que en una primera fase tenga el valor de evaluación más alto. Por ello, se utiliza una estrategia de nicho donde cada *solucion* compite, para mantenerse en la lista de soluciones, con otras parecidas pero no con las que difieren mucho.

Se dice que una *solucion* S es $MejorQue$ una *solucion* $S2$, Ecuación 3.21, si el valor de evaluación de S es mayor que el de $S2$ ó si teniendo el mismo valor de evaluación, la dimensión (número de atributos) de S es menor que el de $S2$.

$$MejorQue(S, S2) = (S.eval > S2.eval) \vee (S.eval = S2.eval \wedge S.dim < S2.dim) \quad (3.21)$$

Se dice que una *solucion* S $Incluye$ una *solucion* $S2$ si el espacio geométrico definido por las reglas de S incluye al definido por $S2$. Por ejemplo, si $S = A_1[0.0 \ldots 0.9]$ y $S2 = A_1[0.0 \ldots 0.4]$, podemos decir que S Incluye $S2$.

Se dice que una *solucion* S es similar a $S2$ en un grado x si el ratio entre el volumen de espacio que se solapa entre S y $S2$ y el volumen total que define S es igual a x.

Si llamamos S a un objeto de tipo $Solucion$, el procedimiento seguido para decidir si añadir o no S, a la lista de soluciones es el siguiente:

1. Si la lista está llena y S es peor que cualquiera de las almacenadas: descartar S.
2. Si alguna *solucion* ya presente en la lista incluye a S y la supera en valor de evaluación: descartar S.
3. Contabilizar cuantas soluciones son similares es más de un 85 % a S.
   - 3.1. Si $similares \leq 3$: si la lista no está llena: añadir S. Si la lista está llena: sustituir la peor *solucion* de la lista por S.
   - 3.2. Si $similares > 3$: Si S es peor que todas las *soluciones* similares: descartar S. Si no, sustituir la peor *solucion*, de entre las similares, por S.

3.8.5.2. Función de evaluación para la fase II.I: $\mathbf{F_3}$

Para la fase II.I de CLIA, se hace uso, como función de evaluación, de $\mathbf{F_3}$, Algoritmo 18. Esta función sigue la estructura general presentada en el Algoritmo 1. Básicamente, la idea es calcular primeramente, los patrones fiables de cada clase, utilizando a continuación solo éstos para evaluar la bondad de cada celda. Dado que esta función va a evaluar celdas calculadas en rejillas de diferente número de dimensiones, se hace necesario establecer una corrección al parámetro $confianza$. Si no se hiciera así, las celdas calculadas en dimensión 3 dominarían totalmente al resto. Téngase en cuenta que el número de celdas a evaluar aumenta en un factor de 35 por dimensión. Así, el proceso de un atributo, en una rejilla de dimensión 1 y $P = 8$, genera 35 hiperceldas a evaluar. En dimensión 2, el número de hiperceldas será de $35 \times 35 = 1225$. A mayor número de celdas a evaluar, mayor probabilidad de encontrar una que presente valores de $\delta densidad$ positivos solo por azar. La corrección de la $confianza$ atendiendo a dicha circunstancia consigue que le probabilidad de que una celda presente valores positivos de $\delta densidad$, solo por azar, sea constante independientemente de la dimensión en la que ha sido generada. Gracias a esta característica el algoritmo está preparado para generar *soluciones* que contengan el número de *reglas* adecuado, evitando especializar éstas de forma excesiva. Por ejemplo: para un problema donde sólo existan dos atributos relevantes, será preferible una solución que abarque estos 2 atributos y sólo éstos.

Dado que queremos optimizar *precision*, la función $\mathbf{F_3}$, una vez calculados los patrones fiables, debe valorar la utilidad que tiene cada celda para optimizar dicha medida. Por otro lado, dado que la siguiente fase del algoritmo va a refinar las soluciones aquí encontradas, conviene disponer de cierta capacidad de refinado futuro de cada *solucion*. En la práctica, esto conlleva mantener cierto número de patrones y no converger demasiado pronto a soluciones precisas pero con pocos patrones. Así, $\mathbf{F_3}$ utiliza, una vez calculados los patrones fiables, la función *F-Score* Ecuación 3.16 para calcular el valor final de evaluación.

Corrección del parámetro $confianza$. El valor de $confianza$ necesario para los cálculos estará en función del parámetro $confClasif$ (parámetro $confianza$ para la fase de clasificación), proporcionado por el usuario de la técnica y la dimensión de la *solucion* a evaluar. El propósito de corregir el valor $confianza$ con un factor dimensional es conseguir que la probabilidad de que una celda presente una valor positivo de $\delta Densidad$, por azar, sea igual independientemente de la dimensión de la rejilla donde haya sido calculada la celda.

Por ejemplo: si trabajamos con un $confClasif = 0.99$, la probabilidad de que una celda, que NO contenga información válida, presente un valor de $\delta densidad > 0$ será de 0,01. Es decir, que de cada 100 celdas no informadas, sólo una será tratada erróneamente como informada. Es decir, el algoritmo será capaz de filtrar adecuadamente 99 celdas, como no informadas, pero aún cometerá un error detectando 1 de las celdas como informada cuando realmente no lo es. Dado que el número de celdas procesadas varía en función de la dimensión de la rejilla, si queremos mantener la probabilidad de error constante para un valor de $confianza$ dado, es necesario corregir ésta aplicando un factor de corrección.

Así, el número de celdas a evaluar en cada dimensión, para $P = 8$, será el mostrado en la Ecuación 3.22.

$$nCeldas(dim) = \underbrace{35^{dim}}_{\text{celdas por rejilla}} \times \underbrace{\binom{nAtrs}{dim}}_{\text{nº rejillas}} \tag{3.22}$$

Tomando como base el número de celdas procesadas en dimensión 1, podemos calcular un factor que compense el diferente número de celdas a evaluar en otras dimensiones.

Este factor se calcula según la Ecuación 3.23.

$$factorDimensional = \frac{nCeldas(dim)}{nCeldas(1)} = \frac{\binom{nAtrs}{dim}}{\binom{nAtrs}{1}} \times 35^{dim-1} \tag{3.23}$$

El valor corregido de $confianza$, para una dimensión dim y un parámetro $confClasif$, vendrá definido por la Ecuación 3.24.

$$confianza = 1 - \left(\frac{1 - confClasif}{factorDimensional}\right) \tag{3.24}$$

El Algoritmo Algoritmo 18, detalla la definición de la función $\mathbf{F_3}$.

Algorithm 18 FUNCTION $\mathbf{F_3}$

Input:

$celda$: una celda de el grid.

$confianza$: valor de confianza.

$total$: agrupación que contiene todos los patrones.

dim: dimensión en la que se ha procesado la agrupación.

$nAtrs$: número de atributos totales del problema.

Output: La función devuelve la F-SCORE fiable estimada.

$densidadTrivial \leftarrow total.C_1 \div (total.C_1 + total.C_0)$
$factorDimensional \leftarrow \frac{\binom{nAtrs}{dim}}{nAtrs} * 35^{dim-1}$
$conf = 1 - \left(\frac{1}{10^{confianza} * factorDimensional}\right)$
$\delta densidad \leftarrow DeltaDensidad(celda.C_1, celda.C_0, conf, densidadTrivial)$
if $\delta densidad > 0$ **then**
    $nt \leftarrow celda.C_1 + celda.C_0$
    $fiable.C_1 \leftarrow nt * (densidadTrivial + \delta densidad)$
    $fiable.C_0 \leftarrow nt - fiable.C_1$
    return $FSCORE(fiable.C_1, fiable.C_0, total.C_1, total.C_0)$
else
    return 0
end if

3.8.6. Fase II.II: Fase agregativa

En esta segunda subfase, se refinan las soluciones anteriores añadiendo, si fuera necesario, más reglas a las soluciones obtenidas en la subfase II.I. Para cada *solucion*

presente en la lista de soluciones, se hace uso de la función Refinar, Algoritmo 19. Dicha función, intenta añadir reglas de forma que la solución vaya mejorando paso a paso. Cuando no encuentra más reglas útiles para añadir, la función retorna con la solución ampliada.

Para encontrar la mejor regla con la que ampliar una solución, la función Refinar, haciendo uso de la función MejoresSoluciones (ver Alg. 11), explora recursivamente una árbol de soluciones hasta una profundidad *dept*. La primera regla, de la mejor *solucion* encontrada a la profundidad máxima de exploración es añadida a la *solucion* que queremos refinar. Como función de evaluación de soluciones se emplea, aquí, $\mathbf{F_4}$, Algoritmo 20.

Algorithm 19 FUNCTION Refinar

Input:

solucion: contiene la *solucion* que queremos refinar.
X: matriz de $nPatrones \times nAtrs$ que contiene el dataset.
Y: vector de $nPatrones$ elementos que contiene la clase de cada patrón.
P: número de intervalos de discretización.
$confianza$: valor de confianza.
depth: profundidad de exploración, número máximo de reglas de cada solución.
nExpand: coeficiente de expansión de cada solución.

Output: La función devuelve *solucion* refinada con todas las reglas que se hayan podido añadir.

```
1: repeat
2:     (XF, YF) ← Seleccionar(X, Y, solucion)
3:     sols ← MejoresSoluciones(XF, YF, P, confianza, depth, nAtrs, nExpand, F_4)
4:     if |sols| > 0 then
5:         sol ← mejorSolucion(sols)
6:         rule ← sol.firstRule()
7:         solucion.rules ← solucion.rules + rule
8:     end if
9: until |sols| = 0
10: return solucion
```

3.8.6.1. Función de evaluación para la fase II.II: $\mathbf{F_4}$

La función $\mathbf{F_4}$, Algoritmo 20, se emplea para evaluar soluciones en la fase II.II. Está basada en la estructura genérica, utilizada en esta tesis, para las funciones de evaluación (ver Alg. 1). Esto es, la función primeramente calcula los patrones fiables y posteriormente utiliza la función *F-Score* para calcular un valor numérico. Como particularidad, para calcular la densidad trivial, la función parte del subespacio definido por la solución a refinar, siendo la densidad trivial la calculada a partir de los patrones de éste y no del total de patrones.

Algorithm 20 FUNCTION $\mathbf{F_4}$

Input:

$celda$: una celda de el grid.

$confianza$: valor de confianza.

$total.C_1, total.C_0$: número de patrones C_1 y C_0, respectivamente, del total de patrones.

$subEspacio.C_1, subEspacio.C_0$: número de patrones C_1 y C_0 del subespacio previo.

Output: La función devuelve F-SCORE fiable estimado.

1: $densidadTrivial \leftarrow (subEspacio.C_1/(subEspacio.C_1 + subEspacio.C_0))$
2: $\delta densidad \leftarrow DeltaDensidad(celda.C_1, celda.C_0, confianza, densidadTrivial)$
3: **if** $\delta densidad > 0$ **then**
4: $\quad nt \leftarrow celda.C_1 + celda.C_0$
5: $\quad fiable.C_1 \leftarrow nt * (densidadTrivial + \delta densidad)$
6: $\quad fiable.C_0 \leftarrow nt - fiable.C_1$
7: $\quad$ **return** $FSCORE(fiable.C_1, fiable.C_0, total.C_1, total.C_0)$
8: **else**
9: $\quad$ **return** 0
10: **end if**

3.8.7. Fase II.III: Consolidación multitarget

Al llegar a esta fase, contamos con un conjunto de soluciones donde cada una de ellas soluciona una zona distinta del problema y lo hace con un grado de *precision* variable.

El objetivo de esta fase es consolidar dicho conjunto de soluciones de forma que obtengamos el conjunto mínimo que optimice el criterio buscado.

Para decidir que soluciones deben permanecer y cuales deben eliminarse, se utilizan los patrones de Entrenamiento y los valores de $\delta densidad$ calculados individualmente para cada *solucion*. El algoritmo intenta eliminar, una por una, cada *solucion* de forma que si los resultados globales (los obtenidos aplicando todas las reglas activas al conjunto de Entrenamiento) no empeoran, la *solucion* puede eliminarse definitivamente. El proceso se repite hasta que ya no pueden eliminarse más soluciones. En ese momento, la lista de soluciones contendrá el conjunto mínimo de reglas que hace máximo el criterio de optimizar.

Aplicando el concepto de fiabilidad y teniendo en mente que el objetivo es optimizar el resultado sobre un conjunto de patrones de Test, desconocido en fase de entrenamiento, a la hora de calcular cuántos patrones son seleccionados por la lista de soluciones, éstos se ajustan utilizando $\delta densidad$ de cada *solucion*.

Los pasos a seguir para llevar a cabo este proceso son:

1. Se ordena, de menor a mayor, la lista de soluciones utilizando como criterio de ordenación la $\delta densidad$ de cada *solucion*.

2. Se evalúa la agrupación, sin tener en cuenta la primera *solucion*. En el Algoritmo 21 se detalla el proceso de evaluación de un conjunto de soluciones. Para evaluar la bondad de una agrupación de soluciones, se calcula la probabilidad de seleccionar cada uno de los patrones del conjunto de Entrenamiento. Una vez calculada ésta, podemos obtener la esperanza de patrones C_1 y C_0 que obtendríamos usando dicho conjunto de soluciones.

3. Si el valor de evaluación de la agrupación, es igual o mejor que el actual, se elimina definitivamente dicha *solucion*.

4. Se repiten los pasos anteriores para todas las soluciones presentes en lista soluciones.

5. Se reitera todo el proceso hasta que no se produzca ninguna baja.

6. Se calculan que soluciones incluyen otras (ver Sección 3.8.5.1) y se eliminan las más especializadas por estar incluidas en otras.

$$poolSoluciones = poolSoluciones - solucion_i \forall solucion_i \subseteq solucion_j \quad (3.25)$$

3.8.7.1. Función de evaluación para Fase II.III: FEvalConjuntoSoluciones

La función de evaluación necesaria para valorar una agrupación de soluciones, FEvalConjuntoSoluciones, se define en el Algoritmo 21. Una vez calculados los patrones C_1 y C_0 esperables al hacer uso de todas las reglas de la lista de soluciones, la función F_5, utilizada por FEvalConjuntoSoluciones, calcula el valor de evaluación conjunto de la agrupación/lista de soluciones.

La función $\mathbf{F_5}$, utilizada en esta tesis, ha sido *precision* (ver Ec. 2.3). Se ha elegido dicha función dado que es *precision* la medida que se ha elegido optimizar en nuestro problema de mercado de valores. Por citar otros ejemplos de funciones que podrían ser de utilidad en el papel de $\mathbf{F_5}$ podríamos nombrar: *recall* (ver Ec. 2.4), *F-Score* (ver Ec. 3.16) y PSR (ver Ec. 3.17).

Algorithm 21 FUNCTION **FEvalConjuntoSoluciones**

Input:

listaSoluciones: lista de soluciones a evaluar.

total: agrupación que contiene todos los patrones.

Output: Valor numérico real.

1: $probs \leftarrow new\ Vector(total.C_1 + total.C_0)$
2: $densidadTrivial \leftarrow \frac{total.C_1}{total.C_1 + total.C_0}$
3: **for** solucion in [listaSoluciones] **do**
4: $probSol \leftarrow \frac{(solucion.deltaDensidad + densidadTrivial) \times (solucion.NC1 + solucion.NC0)}{solucion.NC1}$
5: **for** (patron,i) in [Values] **do**
6: **if** $solucion.regla(patron) = True$ **then**
7: **if** $clase[i] = 0$ **then**
8: $probs[i] \leftarrow 1$
9: **else**
10: $probs[i] \leftarrow probs[i] + probSol$
11: **if** $probs[i] > 1$ **then**
12: $probs[i] \leftarrow 1$
13: **end if**
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **for** i in $[1 .. |Values|$ **do**
19: $nc[clase[i]] \leftarrow probs[i]$
20: **end for**
21: **return** $F_5(nc[1], nc[0], total.C_1, total.C_0)$

4

Validación experimental en problemas Tipo I

Uno de los objetivos de esta tesis consiste en diseñar una técnica que sea capaz de extraer información útil de los datos históricos del mercado de valores, creando modelos que puedan ser utilizados para predecir el comportamiento futuro de éste. En el capítulo de Estado de la Cuestión, se ha hecho una revisión bibliográfica sobre la posible predictibilidad del mercado de valores y se ha visto cómo existe cierta controversia entre si existe o no información utilizable en los datos históricos y, en caso de haberla, se sospecha que será de poca entidad.

Por ello, antes de experimentar con datos reales de bolsa, se han validado las diferentes técnicas en escenarios de problemas sintéticos y se ha estudiado, en profundidad, su sensibilidad a factores que se creen presentes en el mercado de valores, como pueden ser el ruido, la relativamente alta dimensionalidad, la escasez de patrones y la posible autocorrelación de la clase de salida.

Así, primeramente, se ha generado un conjunto de problemas sintéticos. Estos problemas son construidos de forma que estén presentes los factores de dificultad comentados y que permitan estudiar la sensibilidad de cada técnica a cada factor, variando gradualmente la presencia de éstos en cada escenario. Sobre dicho conjunto de problemas, se ha comparado el rendimiento de las diferentes variantes de LIA (LIA, LIA2 y LIA3), entre sí, y frente a tres técnicas estándar de selección de atributos: CFS-Subset, ReliefF y Chi-Squared.

Seguidamente, se ha experimentado con problemas reales generados a partir de

datos públicos del mercado de valores. Sobre éstos, se ha comparado el rendimiento de la nueva técnica de clasificación, CLIA, haciendo uso de LIA como algoritmo de selección de atributos, frente al clasificador Random Forest con Chi-Squared como técnica de selección de atributos.

4.1. Problemas Sintéticos-I

4.1.1. Generación de los problemas Sintéticos-I

Se ha desarrollado una aplicación que, a partir de una definición de problema detallada en lenguaje XML, es capaz de generar ficheros de datos en formatos CVS y/o ARFF adecuados para servir de entrada a diferentes técnicas de Aprendizaje Automático.

En el fichero de definición de dominio, podemos especificar las características de los patrones de datos a generar. El generador ha sido diseñado para trabajar con clases de salida binaria, con dos posibles valores: C_0 y C_1.

A continuación, se detallan las características parámetros que que podemos especificar. Como nomenclatura, se usa $Atributo[k]$, para especificar el valor de un atributo para el patrón k de datos.

- El número de patrones.
- El número de atributos.
- Para cada atributo:
  - El rango (valores mínimo y máximo)
  - La distribución que siguen sus valores.
    - Uniforme: los valores generados siguen una distribución Uniforme.
    - Gausiana: los valores siguen una distribución Gaussiana.
    - Escalonada: el atributo mantiene un valor constante hasta que, aleatoriamente, cambia de valor (ver Fig. 4.1):

$$Atributo[k] = \begin{cases} Atributo[k-1] & \text{con probabilidad } p \\ rnd(0,1) & \text{con probabilidad } 1-p \end{cases}$$

      donde $rnd(0,1)$ es un número aleatorio en el rango $[0,1)$.

- Tendencial: modela atributos del estilo a los que pueden verse en el mercado de valores. Básicamente, cada valor depende del anterior y de unas tendencias o inercias. Se modela para varias escalas de tendencia simulténeas: ciclos pequeños insertos en ciclos amplios (ver Fig. 4.2).

$$Atributo[k] = Atributo[k-1] + Tend_0[k] + Tend_1[k] + Tend_2[k]$$

donde $Tend_i[k]$ vienen dadas por:

$$Tend_i[k] = \begin{cases} Tend_i[k-1] & \text{con probabilidad } p_i \\ rnd(-0.02, 0.02) & \text{con probabilidad } 1 - p_i \end{cases}$$

- La función de salida que se utiliza para generar la clase de cada patrón. La función calcula si un patrón dado, en base al valor de sus atributos, pertenece o no a una región del espacio llamada la región objetivo. En caso de pertenecer, genera la clase C_1 con probabilidad *probin* y C_0 en caso contrario. Si el patrón no pertenece a la región objetivo, genera C_1 con probabilidad *probout*, C_0 en caso contrario.

  Parámetros de la función de salida:

  - *tipo*: establece el tipo de figura geométrica que se usa para definir la región objetivo (ej. paralelepípedo, esfera). Aquí, se han usado regiones objetivo de tipo paralelepípedo.
  - Parámetros específicos dependiendo del tipo de función de salida (ej. radio de la esfera).
  - *probin*: probabilidad de que un patrón que pertenezca a la región objetivo sea de clase C_1.
  - *probout*: probabilidad de que un patrón que no pertenezca a la región objetivo sea de clase C_1. A lo largo de este capítulo, se usan indistintamente los términos *probout* y nivel de ruido.
  - *probautocorrelacion*: presencia de rachas o porcentaje de autocorrelación entre patrones de la clase C_1. Si se activa esta característica el algoritmo genera los patrones de la clase C_1 agrupados en rachas, de la siguiente forma:

$Si(patron[k-1].clase = C_1) -> (patron[k].clase = C_1)$ con $probabilidad = probautocorrelacion$[1]

- *balanceada*: si se activa esta característica, el proceso de generación garantiza que el número de patrones de clase C_1 será igual al de clase C_0.

A modo de ejemplos, la Figura 4.1 muestra los valores generados para un atributo que sigue una distribución Escalonada, generados para 1000 patrones y, la Figura 4.2 muestra un atributo generado según una distribución Tendencial.

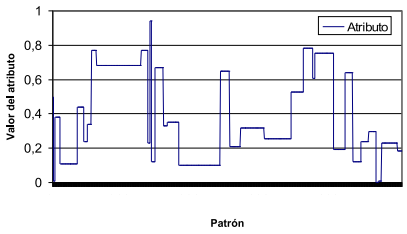

Figura 4.1: Atributo Escalonado

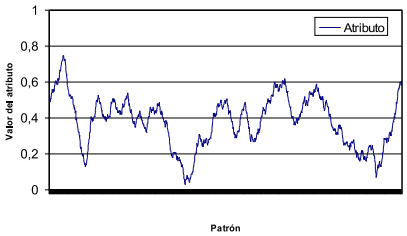

Figura 4.2: Atributo Tendencial

En la Figura 4.3 se muestra, como ejemplo, el espacio tridimensional al que dá lugar un problema con 3 atributos relevantes (X, Y y Z) donde se define una función de salida tipo paralelepípedo. Los puntos azules representan los patrones de clase C_1. Presuponemos en este ejmplo que los patrones de clase C_0 se distribuyen de manera uniforme por todo el espacio. Podemos observar como la densidad de patrones C_1 ($densidadC_1 = C_1/(C_1 + C_0)$), en la región definida por la función de salida, sigue una probabilidad alta (*probin*) y, fuera, una probabilidad más baja (*probout*).

[1]El sistema ajusta dinámicamente los porcentajes de probabilidad de forma que el uso de la autocorrelación no afecte al cómputo global esperable de C_1 y C_0 según los parámetros *probin* y *probout*.

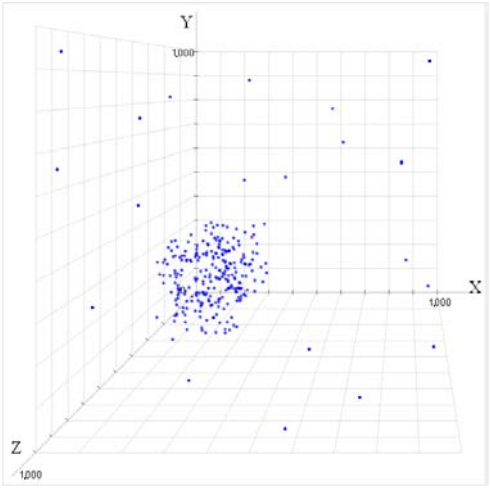

Figura 4.3: Densidad de patrones C_1

4.1.2. Descripción de los problemas Sintéticos-I

Para abarcar un amplio rango de situaciones, 8.160 problemas han sido generados. Se han agrupado en diferentes escenarios con diferentes niveles de dificultad. Para cada combinación de parámetros, se han generado 30 problemas utilizando semillas aleatorias distintas. Con el fin de estudiar el efecto del número de patrones, se han generado conjuntos de problemas con 500, 1000 (1K), 5000 (5K) y 15 000 (15K) patrones. El parámetro *propout* (nivel de ruido), ha tomado los valores: {0.05, 0.10, 0.15} y *propin* (densidad de C_1 en zona objetivo) los siguientes: {0.2, 0.3, 0.4, 0.5, 0.6}.

Como nomenclatura en los problemas Sintéticos, diremos que un problema es de dimensión 1, abreviado como 1D, cuando el número de atributos relevantes que conformen la función de salida sea 1. Procederemos de igual forma para 2D hasta 5D.

A continuación se detallan las características de los diferentes escenarios:

- **Escenario Normal**: cada problema tiene 20 atributos, generados siguiendo una distribución Uniforme. Dependiendo del número de atributos relevantes, se han generado: Normal-1D, Normal-2D, Normal-3D, Normal-4D y Normal-5D con 1, 2, 3, 4, y 5 atributos relevantes, respectivamente.

- **Escenario Grande**: cada problema es creado con 100 atributos, generados siguiendo una distribución Uniforme. Se han generado: Grande-1D, Grande-2D,

Grande-3D, Grande-4D y Grande-5D con 1, 2, 3, 4, y 5 atributos relevantes, respectivamente.

- **Escenario Multizona MT-1**: 20 atributos generados siguiendo una distribución Uniforme. En este escenario, la región objetivo se compone de dos hiperrectángulos, cada uno definido con 3 atributos distintos dando así $3 + 3 = 6$ atributos relevantes. En este caso, $probin = 1.0$ y $probout = 0.0$.

- **Escenarios Multizona MT-2**: 20 atributos generados siguiendo una distribución Uniforme. En este escenario, la región objetivo se compone de dos hiperrectángulos sin solape, definidos ambos con los tres mismos atributos. Así, tenemos tres atributos relevantes.

- **Escenarios Difícil**: 20 atributos generados con distintas distribuciones (Uniforme, Gausiana, Escalonada, Tendencial) y tres atributos relevantes. La clase de salida presenta autocorrelación y los patrones C_1 aparecen en rachas.

En la Tabla 4.1 se muestran las características de los diferentes escenarios, incluyendo el número de patrones generados y el ratio de desbalanceo entre las clases, medido como se propone en (Fernández et al., 2008), $IR = \frac{|C_1|}{|C_0|}$, siendo C_1 la clase minoritaria.

4.1.3. Marco experimental

En esta sección se validan experimentalmente cada una de las variantes de LIA. Para ello, se han utilizado los 8160 problemas que forman el conjunto de problemas llamado Sintéticos-I.

Estos problemas están construidos de forma que estén presentes, en distinto grado, los factores de dificultad (ruido y escasez de patrones), un número variable de atributos totales y diferente grado de dificultad en cada escenario. El objetivo es estudiar la sensibilidad de cada técnica a cada uno de estos factores.

Sobre dicho conjunto de problemas se ha comparado el rendimiento de las diferentes variantes de LIA (LIA, LIA2 y LIA3), entre sí, y frente a tres técnicas estándar de selección de atributos: CFS-Subset, ReliefF y Chi-Squared .

Dado que en este conjunto de problemas el subconjunto de atributos relevantes es conocido *a-priori*, para medir el rendimiento de cada técnica de selección de atributos

Tabla 4.1: Características de los problemas Sintéticos-I

Escenario	Atrs. / A. Relevantes	Distribución	Nº Instancias	IR	Nº Problemas
Normal1D	20 / 1	Uniforme	1K	4.55	450
Normal2D	20 / 2	Uniforme	2K	6.67	450
Normal3D	20 / 3	Uniforme	500, 1K, 5K	8.33	2,700
Normal4D	20 / 4	Uniforme	15K	9.09	450
Normal5D	20 / 5	Uniforme	40K	10.0	450
Grande1D	100 / 2	Uniforme	10K	4.55	450
Grande2D	100 / 2	Uniforme	10K	6.67	450
Grande3D	100 / 3	Uniforme	10K	8.33	450
Grande4D	100 / 4	Uniforme	10K	9.09	450
Grande5D	100 / 5	Uniforme	10K	10.0	450
MT-1	20 / 6	Uniforme	5K	5.26	30
MT-2	20 / 3	Uniforme	5K	5.00	30
Difícil	20 / 3	Uniforme, Gausiana Tendencial, Escalonada	500, 1K, 5K	8.33	1.350

se ha utilizado el *Ratio de éxito* (ver Ec. 4.1), definido como el ratio entre los atributos correctamente seleccionados como relevantes dividido por el número máximo de atributos relevantes realmente contenidos en cada problema.

En las figuras donde se muestran los resultados, se utiliza el promedio de *Ratio de éxito* para el total de problemas que pertenecen a una categoría de problemas dada.

$$Ratio\ de\ éxito\ = \frac{|atributosseleccionados \in atributosrelevantesconocidos|}{100 \times maxAtrs} \quad (4.1)$$

También, se va a estudiar el efecto de diferentes valores del parámetro *depth* (profundidad de exploración), en los resultados de cada variante de LIA .

Como técnicas de selección de atributos con las que comparar se han seleccionado tres algoritmos, incluidos en la herramienta WEKA (Witten et al., 2011): CFS-Subset, ReliefF y Chi-Squared. Puede consultarse una descripción de éstas en el capítulo de Estado de la Cuestión (ver Sección 2.4.1)).Todos los parámetros han tomado el valor por defecto establecido en WEKA. Como algoritmos de búsqueda asociados a CFS-Subset se han utilizado: ExhaustiveSearch para dominios de menos de 25 atributos y BestFirst para el resto.

Por último, LIA se ha configurado para estos experimentos haciendo uso del parámetro $mode = 1$. Este modo de funcionamiento se ha diseñado para encontrar la zona de máxima densidad de patrones fiables de tipo C_1. Una descripción más extensa de dicho parámetro puede encontrarse en la Sección 3.3.2.

4.1.4. Resultados experimentales en problemas Sintéticos-I

A continuación, se van a presentar los resultados obtenidos por las variantes de LIA para selección de atributos: LIA, LIA2 y LIA3. Se van a establecer comparaciones entre ellas y frente a CFS-Subset, ReliefF y Chi-Squared. En la Tabla 4.2 se muestran los resultados obtenidos con estas últimas sobre el conjunto de problemas Sintéticos-I. Como puede observarse, Chi-Squared ha obtenido en promedio los mejores resultados, seguida de CFS-Subset y ReliefF . Observando el comportamiento, escenario a escenario, también se constata como Chi-Squared proporciona resultados mejores en escenarios 1D, 2D, 3D y **MT**. Para los escenarios 4D y 5D la técnica CFS-Subset proporciona los mejores resultados. En el escenario **Difícil** el mejor resultado lo obtiene CFS-Subset seguido de CFS-Subset y ReliefF. Por último, ReliefF proporciona los mejores resultados en escenarios **MT**.

Tabla 4.2: *Ratio de éxito* con CFS-Subset, ReliefF y Chi-Squared

Escenario	CFS-Subset	ReliefF	Chi-Squared
Normal1D	99 %	98 %	97 %
Grande1D	100 %	100 %	100 %
Grande2D	96 %	95 %	100 %
Normal2D	93 %	94 %	93 %
Grande3D	85 %	72 %	89 %
MT-1	72 %	88 %	100 %
MT-2	64 %	92 %	96 %
Normal4D	67 %	64 %	66 %
Normal5D	47 %	40 %	43 %
Normal3D	55 %	63 %	53 %
Grande4D	52 %	25 %	46 %
Difícil	32 %	22 %	38 %
Grande5D	13 %	9 %	7 %
Promedio	**67,3 %**	**66,3 %**	**71,4 %**

En la Tabla 4.3 se muestran los tiempos necesarios para completar el juego de pruebas completo. De las tres técnicas, Chi-Squared ha sido la técnica más rápida, seguida por `CFS-Subset`. ReliefF ha tardado mucho más tiempo en completar todas las pruebas, 21 días, frente a las 22 horas de Chi-Squared.

Tabla 4.3: Tiempos de procesamiento con CFS-Subset, ReliefF y Chi-Squared

CFS-Subset	ReliefF	Chi-Squared
37 horas	21 días	22 horas

4.1.4.1. Resultados experimentales con LIA

En la Tabla 4.4 se muestran los resultados obtenidos con LIA para distintos valores del parámetro *depth* (profundidad de exploración). *Opt* indica la utilización del mecanismo de optimización que consiste en procesar solo las combinaciones de tres atributos donde existan al menos dos con información. (ver Alg. 3.3.2). Como puede observarse, a mayor profundidad de exploración mejores resultados (*Ratio de éxito*) se obtiene.

Tabla 4.4: *Ratio de éxito* con LIA

Escenario	$(depth = 1)$	$(depth = 2)$	$(depth = 3, opt)$	$(depth = 3)$
Normal1D	100 %	100 %	100 %	100 %
Grande1D	100 %	100 %	100 %	100 %
Grande2D	100 %	100 %	100 %	100 %
Normal2D	98 %	100 %	100 %	99 %
Grande3D	94 %	96 %	96 %	98 %
MT-1	99 %	99 %	99 %	98 %
MT-2	93 %	99 %	99 %	100 %
Normal4D	84 %	91 %	92 %	93 %
Normal5D	65 %	76 %	80 %	81 %
Normal3D	68 %	75 %	76 %	77 %
Grande4D	61 %	71 %	75 %	78 %
Difícil	44 %	46 %	46 %	46 %
Grande5D	20 %	23 %	27 %	31 %
Promedio	**78,92 %**	**82,77 %**	**83,85 %**	**84,65 %**

En la Tabla 4.5 se muestran los tiempos de LIA, para cada valor de *depth*. Como puede observarse, el parámetro *depth* condiciona en gran medida el tiempo de proceso. A mayor profundidad de exploración mayor tiempo de proceso. A la hora de establecer comparaciones debe considerarse que el tiempo de carga de los problemas y el tiempo de la discretización previa, común a todos los valores de *depth*, asciende a unos 5 minutos. En problemas con pocos atributos, estos tiempos pueden llegar a ser muy significativos respecto al tiempo total de proceso.

Observando los resultados de LIA($dept = 3, opt$), en las Tablas 4.4 y 4.5, podemos ver como gracias a esta optimización obtenemos resultados notablemente mejores que para $depth = 2$ sin incurrir en mucho mayor tiempo de proceso.

Tabla 4.5: Tiempos de procesamiento con LIA

LIA($dept = 1$)	**LIA**($dept = 2$)	**LIA**($dept = 3, opt$)	**LIA**($dept = 3$)
7 minutos	30 minutos	45 minutos	7 horas

Frente a estos tiempos, la Tabla 4.3 enseña como Chi-Squared ha tardado 22 horas, CFS-Subset 37 horas y ReliefF 21 días. No obstante, la comparación de tiempos entre WEKA y LIA, debe tomarse con cierta prudencia, dado el diferente lenguaje de implementación (C++ *vs* JAVA) y los diferentes modos de invocación o llamada para cada problema. En cuanto a consumo de recursos, durante el proceso LIA ha consumido un máximo de 3MB de memoria, frente a 1.5 GB de WEKA.

En la Figura 4.4 se compara LIA, para $depth = 3$, frente a CFS-Subset, ReliefF y Chi-Squared. Como puede observarse, LIA($depth = 3$) muestra resultados superiores al resto de técnicas. Esta diferencia positiva aumenta según se incrementa la dificultad del escenario utilizado, especialmente, para escenarios de dimensión 3, 4 ó 5.

En la Figura 4.5 se muestra la mejora porcentual del promedio de *Ratio de éxito* para cada nivel de ruido, *probout*, obtenida al utilizar LIA(3D) frente a cada una de las técnicas analizadas. Como puede observarse, LIA siempre supera al resto de técnicas, para todos los niveles de ruido analizados. Además, cuanto mayor es el nivel de ruido del problema, más favorable es la comparación de LIA frente al resto de técnicas. Puede observarse, como para valores de ruido del 5 %, LIA mejora al resto de técnicas entre un 25 % y un 38 %. La mejora máxima se observa para los valores de ruido más altos, 15 %, en ellos, LIA supera al resto entre un 45 % y un 80 %.

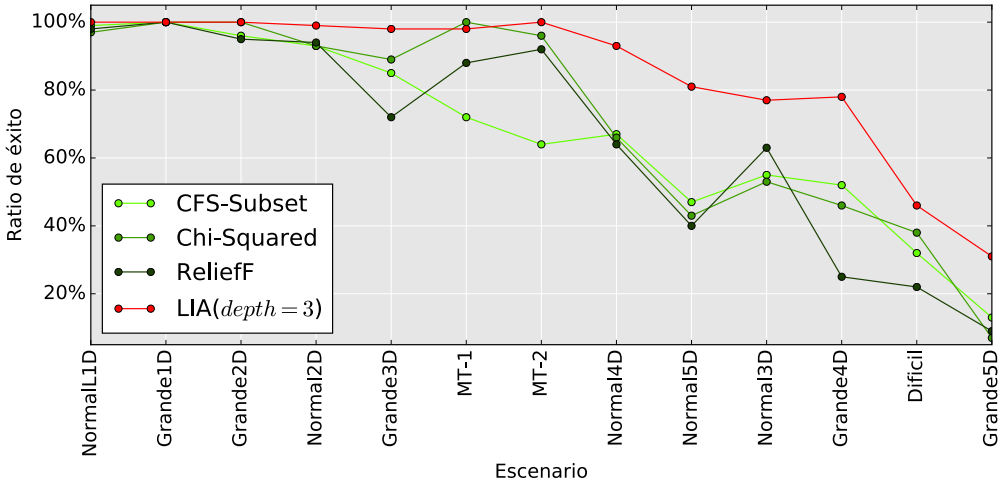

Figura 4.4: LIA($depth = 3$) *vs* CFS-Subset, ReliefF y Chi-Squared

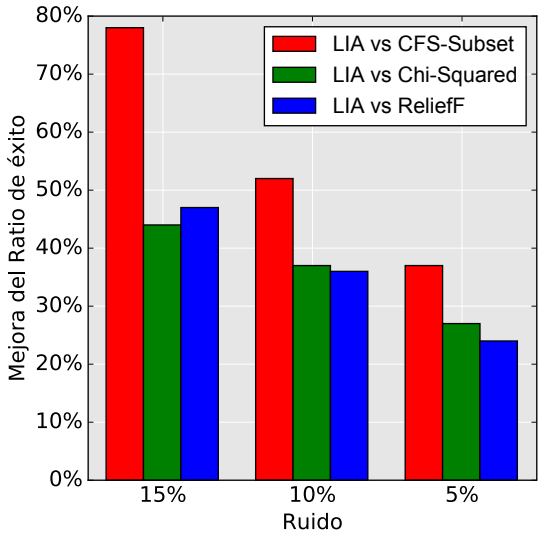

Figura 4.5: Sensibilidad a ruido

En la Figura 4.6 se muestra un estudio similar, en este caso sobre la sensibilidad al número de patrones. Se muestra el porcentaje de mejora sobre el promedio de *Ratio de éxito* de LIA *versus* CFS-Subset, ReliefF y Chi-Squared para problemas de 500, 1000 (1K) y 5000 (5K) patrones, respectivamente. Como se observa en la figura, LIA supera al resto de técnicas en todos los casos. Para 5000 patrones, la mejora va del 20 % al

40 %. Este diferencial aumenta según disminuye el número de patrones, llegando a un 40 % a 140 % para 500 patrones. Así, puede verse como LIA es menos sensible a la escasez de patrones que el resto de técnicas analizadas.

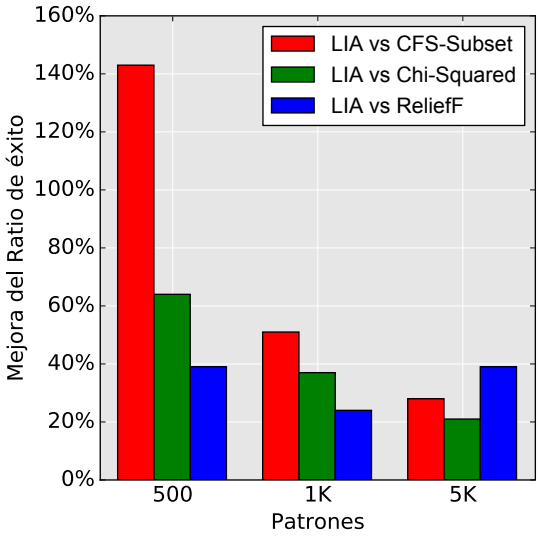

Figura 4.6: Sensibilidad a número de patrones.

4.1.4.2. Resultados experimentales con LIA2

Siguiendo el mismo procedimiento que para LIA, se ha procedido a experimentar con LIA2. En la Tabla 4.6 se muestran los resultados obtenidos con LIA2 para *depth*={*1, 2 y 3*}. Como puede observarse, los resultados van mejorando progresivamente según aumenta la profundidad de exploración (*depth*). Siendo los obtenidos con $depth = 3$ los mejores, tanto en promedio como escenario a escenario, con la única excepción de **MT-1** donde $depth = 2$ ha obtenido un mejor resultado.

En la Figura 4.7 se compara el rendimiento de LIA2, para $depth = 3$, frente a: CFS-Subset, ReliefF y Chi-Squared. Como puede constatarse, LIA2 supera al resto de técnicas en todos los escenarios, aumentando la diferencia positiva según avanzamos en grado de dificultad del escenario. La diferencia es especialmente llamativa para escenarios **Normal4D** y **Normal5D**.

En la Tabla 4.7 se muestra el tiempo de procesamiento que ha requerido LIA2 para completar el experimento, para cada valor del parámetro *depth*. Desde 9 minutos, para

Tabla 4.6: *Ratio de éxito* con LIA2

Escenario	$(depth = 1)$	$(depth = 2)$	$(depth = 3)$
Normal1D	100 %	100 %	100 %
Grande1D	100 %	100 %	100 %
Grande2D	100 %	100 %	100 %
Normal2D	98 %	99 %	99 %
Grande3D	93 %	96 %	96 %
MT-1	93 %	96 %	94 %
MT-2	84 %	97 %	97 %
Normal4D	84 %	89 %	91 %
Normal5D	65 %	76 %	79 %
Normal3D	69 %	74 %	75 %
Grande4D	62 %	71 %	73 %
Difícil	38 %	39 %	39 %
Grande5D	20 %	22 %	23 %
Promedio	**77,4** %	**81,5** %	**82,0** %

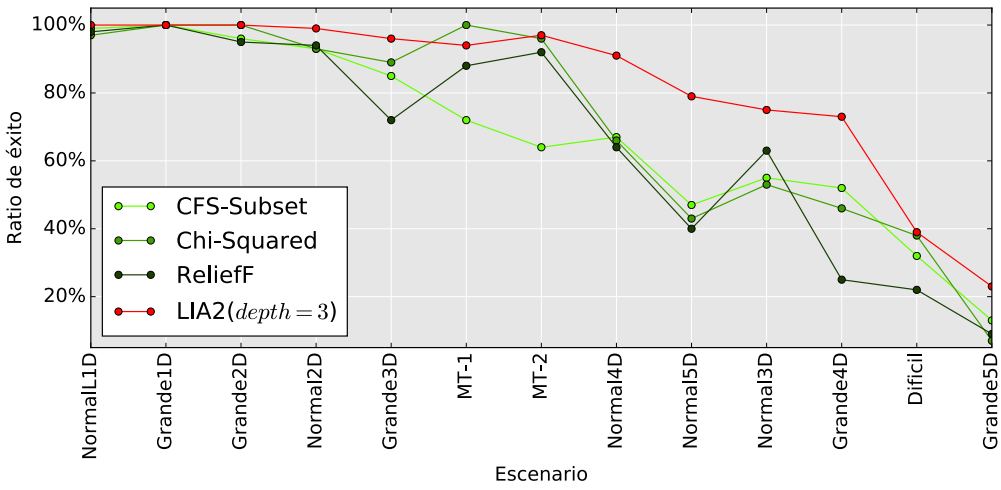

Figura 4.7: LIA2$(depth = 3)$ *vs* CFS-Subset, ReliefF y Chi-Squared

$depth = 1$, hasta 51 minutos para $depth = 3$. En cuanto a consumo de recursos, durante el proceso LIA2 ha consumido un máximo de 9MB de memoria.

Tabla 4.7: Tiempos de procesamiento con LIA2

$(depth = 1)$	$(depth = 2)$	$(depth = 3)$
9 minutos	17 minutos	51 minutos

4.1.4.3. Resultados experimentales con LIA3

Siguiendo el mismo procedimiento experimental que para LIA y LIA2, se ha realizado la experimentación con LIA3. En la Tabla 4.8 se muestran los resultados obtenidos con LIA3 para *depth*=*{1, 2 y 3}*. Los resultados progresivamente van mejorando según aumenta *depth*.

En la Figura 4.8 se compara el rendimiento de LIA3, para $depth = 3$, frente a CFS-Subset, ReliefF y Chi-Squared. Puede observarse como LIA3 supera al resto de técnicas en todos los escenarios, haciendose máxima la diferencia para escenarios de 3, 4 y 5 dimensiones.

Tabla 4.8: *Ratio de éxito* con LIA3

Escenario	$(depth = 1)$	$(depth = 2)$	$(depth = 3)$
Normal1D	100 %	100 %	100 %
Grande1D	100 %	100 %	100 %
Grande2D	100 %	100 %	100 %
Normal2D	98 %	98 %	98 %
Grande3D	95 %	95 %	96 %
MT-1	93 %	93 %	95 %
MT-2	96 %	98 %	98 %
Normal4D	91 %	91 %	93 %
Normal5D	79 %	82 %	85 %
Normal3D	71 %	72 %	74 %
Grande4D	73 %	76 %	78 %
Difícil	41 %	41 %	41 %
Grande5D	23 %	29 %	29 %
Promedio	**81,54 %**	**82,69 %**	**83,6 %**

En la Tabla 4.9 se muestra el tiempo de procesamiento que ha requerido LIA3 para completar el experimento, para diferentes valores del parámetro *depth*. LIA3 ha completado el test en 17 minutos, para $depth = 1$. Utilizando $depth = 3$, los tiempos de

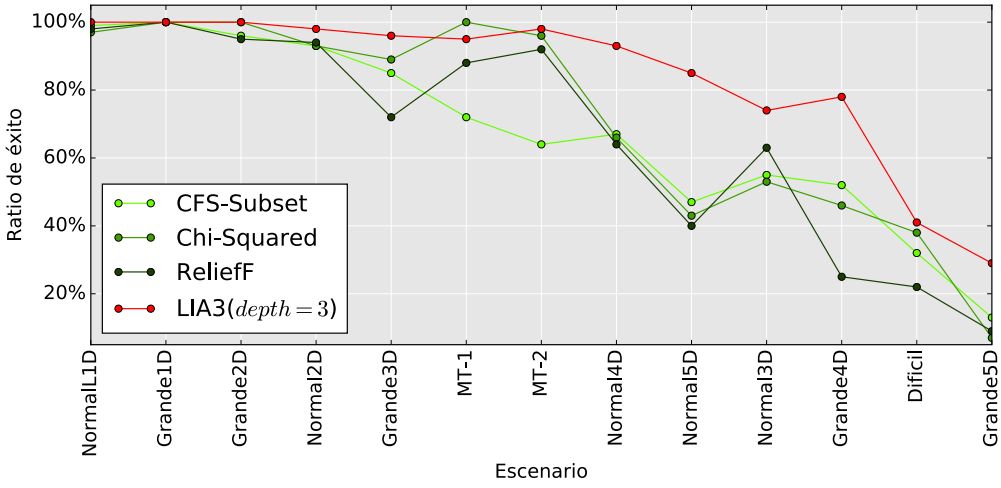

Figura 4.8: LIA3($depth = 3$) *vs* CFS-Subset, ReliefF y Chi-Squared

proceso se incrementan hasta 143 minutos. En cuanto a consumo de recursos, durante el proceso LIA3 ha consumido un máximo de 29MB de memoria.

Tabla 4.9: Tiempos de procesamiento con LIA3

($depth = 1$)	($depth = 2$)	($depth = 3$)
17 minutos	41 minutos	143 minutos

4.1.4.4. Comparación entre LIA, LIA2 y LIA3

En la Figura 4.9 se compara el promedio de *Ratio de éxito* obtenido por LIA, LIA2 y LIA3, para el total de problemas que componen el conjunto de pruebas Sinteticos-I y para cada valor del parámetro *depth*.

Las diferencias, entre unas y otras, no son excesivamente grandes. Pero sí puede apreciarse como LIA2 es, para todos los valores de *depth*, la técnica menos precisa. LIA es la que obtiene mejores resultados para $depth > 1$. Siendo LIA3 la mejor en $depth = 1$ seguida de LIA. No obstante, nótese que mientras que en LIA y LIA2, el parámetro *depth* establece la dimensión de la solución encontrada, por ejemplo, $depth = 2$ evalúa y encuentra soluciones de 2 atributos, en el caso de LIA3 el parámetro *depth* establecece la dimensión de las soluciones parciales evaluadas pero no así del árbol completo de soluciones. Así, aún explorando el espacio de soluciones para $depth = 1$ LIA3 proporcionaría

árboles de dimensiones o profundidades arbitrariamente complejas. Por ello, atendiendo a la complejidad de la solución encontrada (número de atributos), LIA3($depth = 1$) es comparable con LIA y LIA2 para $depth = 3$.

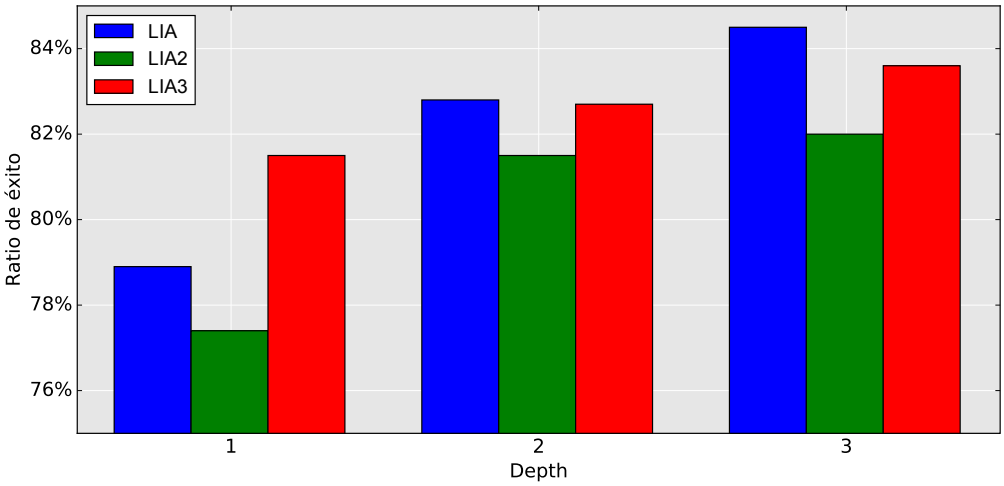

Figura 4.9: Comparativa LIA, LIA2 y LIA3

Para todas las técnicas y para todos los valores de *depth*, puede observarse como el *Ratio de éxito* mejora según aumentamos el valor de *depth*. Por ello, los mejores resultados se obtienen en todas las técnicas para $depth = 3$.

En la Figura 4.10 se muestran los resultados para todas las variantes de LIA haciendo uso de $depth = 3$. Como puede observarse, el desempeño de todas ellas está bastante igualado. Si bien, las diferencias entre unas y otras son pequeñas, puede observarse como LIA supera al resto en prácticamente todos los escenarios. LIA3 presenta un resultado intermedio. Y, LIA2 presenta el peor desempeño. Resulta reseñable, también, el resultado obtenido para el escenario **Difícil**: LIA ha obtenido un 46 % de *Ratio de éxito* , frente a 39 % de LIA2 y 41 % de LIA3.

En la Figura 4.11 se muestran los tiempos de procesamiento que ha requerido cada variante, para cada valor de *depth*. Puede observarse como, en todos los casos, el tiempo de proceso aumenta según se incrementa *depth*. LIA ha sido la técnica más rápida para $depth = 1$, ha tenido un comportamiento intermedio para $depth = 2$ y ha sido, con diferencia, la más lenta para $depth = 3$. LIA2 ha presentado un comportamiento mucho más estable aumentando casi linealmente el tiempo de proceso en relación a *depth*. Y LIA3 ha presentando un comportamiento intermedio entre LIA y LIA2.

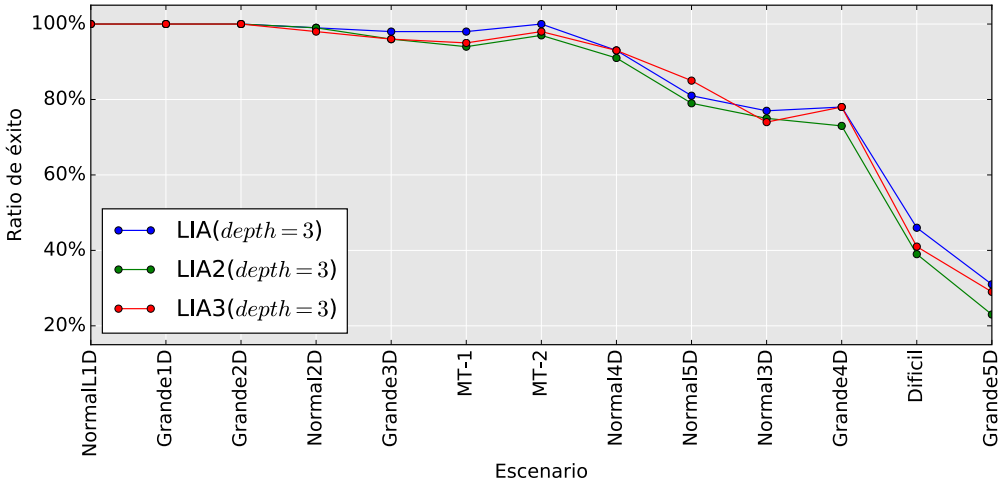

Figura 4.10: Comparativa LIA, LIA2 y LIA3 para $depth = 3$

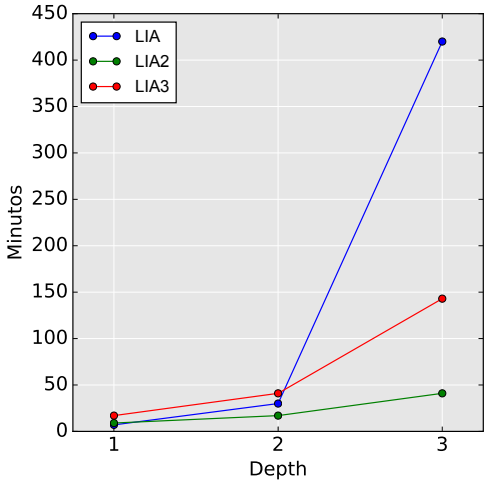

Figura 4.11: Comparativa de tiempos con LIA, LIA2 y LIA3

Dado que se ha visto como el *Ratio de éxito* mejora, en todas las variantes, según se aumenta el parámetro *depth* y, por otro lado, se ha visto como el aumento de *depth* produce un aumento del tiempo de proceso, se puede concluir que las mejoras de *Ratio de éxito* se obtienen en todos los casos a expensas de un mayor tiempo de proceso.

Así, el uso en la práctica de uno u otro valor de *depth* sería una decisión a tomar en base al tamaño del problema y al tiempo máximo aceptable para obtener un resultado.

4.1.5. Resultados experimentales en Sintéticos Especiales

Tal como se explica en mayor detalle en la Sección 3.1.3, los algoritmos que implementan búsquedas *greedy* requieren que el problema a resolver presente algún grado de monoticidad. Es decir, dado que estos algoritmos van construyendo la solución final en base a refinar, paso a paso, soluciones parciales es necesario que dichas soluciones presenten señales/información que indique el camino a seguir hasta la solución global. Si un problema dado no presenta monoticidad a dimensiones bajas, por ejemplo, porque presenta símetrías en los valores de los atributos relevantes, los métodos *greedy* ven degradado su rendimiento hasta el punto de no ser capaces de encontrar las soluciones correctas.

Debido a que LIA explora soluciones en 1, 2 ó 3 dimensiones debería ser capaz, *a-priori*, de enfrentar con garantías este tipo de problemas. LIA2, LIA3, CFS-Subset y Chi-Squared pueden tener más dificultades para encontrar información en estos problemas.

El objetivo de este experimento es validar si efectivamente esto es así. Para ello se ha generado un pequeño conjunto de problemas sintéticos de prueba. Dichos problemas presentan ciertas simetrías que perjudican o impiden el tratamiento correcto en una sola dimensión. Todos los problemas comparten la siguiente característica: es posible encontrar dos atributos que resuelvan de forma óptima el problema. Es decir, existen dos atributos que conjuntamente permiten separar los patrones de cada clase con una precisión del 100 %.

Para cada escenario se han generado 30 problemas con semillas aleatorias diferentes. La elección de los dos atributos relevantes se realiza de forma también aleatoria para cada problema. En todos los casos, cada problema esta formado por 2000 patrones.

A continuación se detallan los escenarios que se han generado:

- **Especial-O**: cada patrón está formado por 100 atributos generados siguiendo una distribución Uniforme. Este escenario no presenta simetrías por lo que todos los problemas deberían ser resueltos sin mayor dificultad por todos los métodos de selección de atributos probados aquí.

- **Especial-A**: en este caso, se han creado 100 atributos binarios, que toman valores aleatorios $\{0, 1\}$. Este escenario presenta simetrías, tipo mosaico, en los atributos relevantes.

- **Especial-B**: 100 atributos, con números reales, generados siguiendo una distribución Uniforme. Este escenario también presenta simetrías, tipo mosaico, en los atributos relevantes.

En la Figura 4.12 se muestra la distribución de patrones de cada clase (rojo y azul), mapeada en los valores de entrada que tienen los dos atributos relevantes.

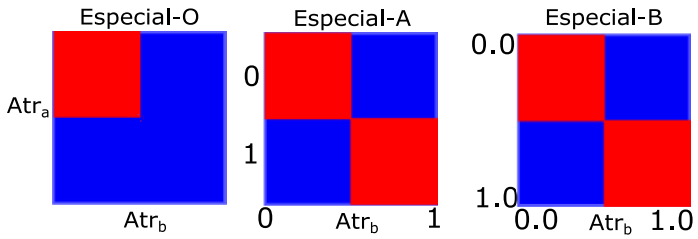

Figura 4.12: Sintéticos especiales

En la Tabla 4.10 se muestran los resultados obtenidos con cada una de las técnicas. A cada técnica se le ha solicitado seleccionar un máximo de dos atributos. Los valores representan el número de problemas donde, entre los dos seleccionados, estaban los dos atributos realmente relevantes. Como puede observarse, todas las técnicas resuelven el 100 % de los problemas de tipo **Especial-O**. LIA para $depth > 1$ resuelve, también, el 100 % de los escenarios A y B. LIA2 y LIA3, no consiguen resolver adecuadamente los escenarios **Especial-A** y **Especial-B**, más allá de unos pocos problemas. LIA3 parece resolver algunos problemas más que LIA2, pero en todo caso muy alejada de los resultados de LIA. CFS-Subset y Chi-Squared no resuelve ningún problema de los escenarios A y B. Por último, ReliefF resuelve bastantes problemas de tipo A y B, 28 y 22, respectivamente.

Estos resultados parecen consistentes con el proceso de exploración que hace cada técnica. Los escenarios **Especial-A** y **Especial-B**, no se resuelven con técnicas que observan los datos solo en una dimensión, dado que no existe ningún atributo que aporte, por separado, información sobre la clase de salida, como es el caso de LIA2, LIA3, CFS-Subset y Chi-Squared .

Tabla 4.10: Comparativa en Sintéticos Especiales

Técnica	Especial-O	Especial-A	Especial-B
LIA($depth = 1$)	30	0	0
LIA($depth = 2$)	30	30	30
LIA($depth = 3$)	30	30	30
LIA2($depth = 1$)	30	0	0
LIA2($depth = 2$)	30	2	1
LIA2($depth = 3$)	30	6	5
LIA3($depth = 1$)	30	4	2
LIA3($depth = 2$)	30	8	5
LIA3($depth = 3$)	30	7	5
CFS-Subset	30	0	0
ReliefF	30	28	22
Chi-Squared	30	0	0

4.1.6. Resumen de los resultados en Sintéticos-I

En este bloque experimental, correspondiente a problemas Sintéticos-I, se han ido valorando las diferentes variantes de LIA desarrolladas frente a tres técnicas comunes de selección de atributos: CFS-Subset, ReliefF y Chi-Squared. Dado que los atributos relevantes en estos problemas son conocidos *a-priori* es posible evaluar el desempeño de cada técnica de forma directa utilizando como medida *Ratio de éxito*. Se ha estudiado el desempeño de cada técnica frente al grado de dificultad de cada escenario y su robusted frente al ruido y a la escasez de patrones.

Para todos los escenarios planteados, todas las variantes de LIA (LIA, LIA2 y LIA3) han obtenido resultados superiores a las técnicas utilizadas para comparar. Esa superioridad se evidencia en mayor medida para escenarios de grado creciente de dificultad y para situaciones de escasez de patrones y alta incidencia de ruido. Frente a LIA, CFS-Subset se ha mostrado en estos escenarios como la técnica menos precisa. ReliefF y Chi-Squared han obtenido resultados similares, ligeramente a favor de Chi-Squared, si bien, Chi-Squared es, además, mucho más rápido.

En cuanto a la comparación entre las diferentes variantes de LIA, podemos resumir que cada una tiene sus puntos fuertes y sus puntos débiles: LIA por su carácter exhaustivo y gran redundancia, obtiene excelentes resultados. Por contra, escala con

dificultad para valores de $depth > 1$. LIA2, es una implementación de compromiso, razonablemente rápida y sencilla pero obtiene, en general, peores resultados que LIA. Por último, LIA3, obtiene unos resultados en gran parte comparables a LIA, aunque peores en el escenario Difícil.

Para todas las variantes de LIA se observa que mejoran los resultados según aumenta *depth*.

En cuanto a los tiempos de proceso, según aumenta la profundidad de exploración, así aumenta el tiempo de proceso. El impacto de este factor es mucho más relevante para LIA y menos para LIA2 y LIA3. Estos últimos algoritmos no exploran exhaustivamente todas las combinaciones de atributos, apostando solo por las, previsiblemente, más prometedoras.

En base a todo ello, podría recomendarse el uso de LIA para problemas de Tipo-I utilizando $depth = 1$ y utilizar $depth = 2$ solo para problemas de menos de unos miles de atributos. El uso de $depth = 3$ se reservaría a problemas con menos atributos y una presencia alta de ruido. El uso de LIA2 y LIA3 en problemas Tipo-I sería recomendable para cualquier tipo de problema independientemente del número de atributos.

El escenario Difícil se ha diseñado teniendo en mente las características que se piensa pueden estar presentes en el mercado de valores. Dado que LIA ha sido la variante más eficaz en estos problemas, por delante de LIA2 y LIA3, se considera que LIA podría ser la más efectiva en dichos problemas reales.

4.2. Mercado de valores

Una vez se han validado las diferentes variantes de LIA en escenarios sintéticos y se ha estudiado su sensibilidad a diferentes factores, se ha procedido a experimentar sobre problemas reales del mercado de valores.

En esta sección, se describen la nomenclatura y los datos utilizados, el pretratamiento de los mismos, la metodología experimental y los resultados obtenidos en dichos problemas.

Como técnica de selección de atributos se ha ha elegido LIA frente a LIA2 y LIA3, a partir de los experimentos realizados sobre problemas sintéticos, donde LIA ha sido la variante técnica que ha obtenido mejores resultados, especialmente en los escenarios llamados Difíciles. Como clasificador se ha usado CLIA (ver Sección 3.8).

Para confrontar CLIA se ha usado Random Forest (ver Sección 2.4.2), precedida de una fase de selección de atributos realizada con Chi-Squared (ver Sección 2.4). El motivo para elegir Random Forest ha sido la eficacia general que presenta esta técnica, incluso en presencia de ruido, y los buenos precedentes publicados (ver Sección 2.1.1) en el campo del Aprendizaje Automático aplicado al mercado de valores. Se ha elegido Chi-Squared dado que ha sido la técnica de selección de atributos que mejores resultados ha proporcionado en los problemas sintéticos, por detrás de LIA.

Los experimentos se han dividido en dos bloques llamados: Bolsa-I y Bolsa-II. En Bolsa-I se ha estudiado la capacidad de cada técnica para aprovechar un hipotético atributo informado escondido entre el resto de atributos reales utilizados en el experimento. En Bolsa-II se ha experimentado exclusivamente con datos reales, con la finalidad de dilucidar si en éstos existe información y cual de las técnicas utilizadas ha sido más eficaz.

4.2.1. Nomenclatura

A continuación se van a describir los términos técnicos usados a lo largo de capítulo. Muchos de ellos son de procedencia anglosajona y de uso común en este área.

- FTSE100: índice de las 100 mayores empresas que cotizan en la bolsa de Londres. Cada mes se revisan, por lo que salen empresas del índice y entran nuevas.
- En este capítulo, se usan indistintamente las palabras: valor, acción y empresa.

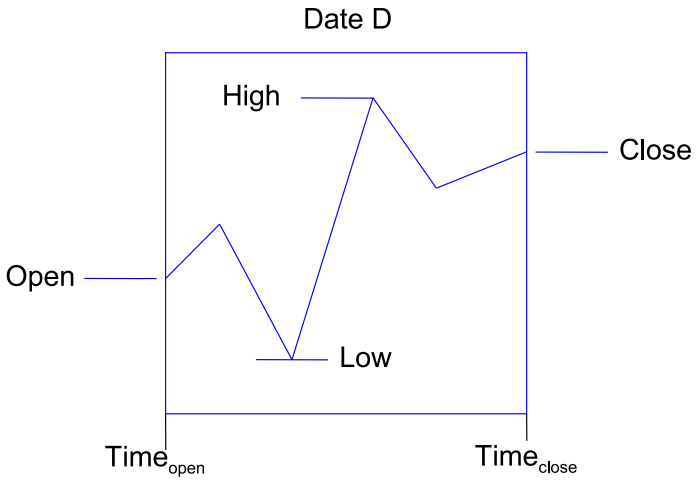

Figura 4.13: Campos de un *Tick Data*

- *Tick Data*: los mercados de valores suministran a través de proveedores de datos, información sobre la operativa del mercado. Dependiendo del coste de la misma, se puede acceder a información en tiempo real (la más cara), información retrasada 15 minutos, información a final de día (habitualmente gratuita) y/o información histórica. Para cada valor cotizado y período de cotización, se suministran varios campos de información. Los más básicos son (se mantienen los nombres en inglés dado que son así como son conocidos en los servicios proveedores de datos y en las bases de datos históricas): *open*, *high*, *low*, *close*, *volume*, *bid*, *ask*. Cada campo se representa con un número real. En la Figura 4.13 se muestran gráficamente los campos principales de la cotización de un valor para un período dado.

  - *Open* es el precio utilizado en la primera operación realizada en el mercado en un período dado. Es decir, representa el primer cruce entre el precio por el que alguien ha estado dispuesto a vender una acción y el precio que alguien ha estado dispuesto a pagar por la misma.
  - *Close* es el precio del último cruce de precios.
  - *High* es el precio más alto que ha alcanzado la cotización de la acción.
  - *Low* es el precio más bajo.
  - *Volume* es el número de acciones intercambiadas en dicho período.
  - *Bid* es el precio medio ofertado para las operaciones de venta.

- *Ask* es el precio medio ofertado para las operaciones de compra.

Así, los 5 primeros campos ($open, high, low, close$, $volume$) representan operaciones realmente completadas en el mercado. Por el contrario, los dos últimos (bid, ask) representan ofertas o deseos de los compradores y vendedores.

- En este capítulo se usará la siguiente nomenclatura en relación a cómo definir el acceso a cada campo de información:
  - Para datos diarios: TicksData[empresa E,día D]. Donde E es el nombre de la empresa cotizada y D, podrá expresarse como una fecha o, alternativamente, como un número entero que indica el número de días desde el primer día del experimento(1/2/2006). Así, índice día 3, equivaldrá a la fecha 4/2/2006. Ejemplo $TicksData[IBM, 4/2/2006].close$ es equivalente a $TicksData[IBM, 4].close$.
  - Para datos intradiarios: $TicksData[EmpresaE, DiaD, MinutoM]$. Ejemplo $TicksData[IBM, 4/2/2006, 510].close$ es el valor del campo *close* de las acciones de IBM, para el día 4/2/2006 en el minuto 510 de la sesión (17:30h).
- Decimos que disponemos de datos minutales si contamos con datos de cotización para cada minuto de los que ha estado abierto el mercado. Los intervalos de tiempo más comunes son: minutales, 5 minutos, 15 minutos y/o diarios.
- Precio(empresa E, día D): el precio de las acciones de una empresa E para el día D, será el importe del último cruce de precios del día D. Así, $Precio(E, D) = Precio(E, D, M)$, donde M indica minuto y vale 510 (último minuto de mercado).

  En general, $Precio(E, D, M) = TicksData[E, D, M].close$ pero, para acciones poco líquidas (empresas pequeñas que tienen un bajo volumen de intercambio de acciones al día), puede darse el caso de que en un minuto dado no se hayan completado operaciones de compra/venta. En ese caso, si $TickData[E, D, M]$ no es conocido, entonces $Precio(E, D, M) = TicksData(E, D, m).close$, donde m sería el primer minuto con operaciones de compra/venta y posterior a M, para la empresa E. Si no existiera ningún minuto posterior a M con cotización conocida, m tomaría el valor del último minuto con cotización conocida para la empresa E el día D.

- Ventana histórica: días y/o minutos del pasado con los que se realiza el entrenamiento de nuestra técnica para predecir el futuro.

- Indicador técnico: fórmula matemática que, a partir de datos históricos, calcula un valor, generalmente un número real, que puede aportar información para operar en bolsa. Se han desarrollado una gran variedad de indicadores. En el capítulo de Estado de la Cuestión, en la Sección 2.1.2, se aporta una enumeración de indicadores, organizados por familias.

4.2.2. Descripción de los datos

Los datos utilizados para realizar este experimento abarcan del año 2006 al 2013, tienen frecuencia minutal (510 *tick data* por día) y cubren el total de empresas que conforman y han conformado el FTSE100.

El proceso de recolección y limpieza de datos, el pretratamiento de los mismos (correcciones de *splits*) y la propiedad de la base de datos resultante, pertenece a la empresa Schulenburg Capital Limited (*https://www.schulenburgcapital.com/*), especializada en la aplicación de técnicas de aprendizaje automático a la gestión de fondos de inversión en el mercado de valores.

El número total de empresas es de 173, si bien, para cada día concreto, sólo 100 de éstas han formado el índice FTSE100. La inclusión del 100 % de empresas en este estudio tiene como objetivo no caer en el Sesgo de Supervivencia (conocido en la literatura como *Survivorship bias*)[1].

El número total de días cotizados, en dicho período, ha sido de 1.934 días.

La generación del conjunto de atributos que define cada instante(día) de bolsa para cada empresa cotizada, se lleva a cabo siguiendo un flujo de dos etapas tal como se muestra en la Figura 4.14.

Partiendo de la base de datos que contiene los *ticks data*, en una **primera etapa** se generan para cada empresa y día, un conjunto de indicadores técnicos, intentando abarcar los más habituales y tratando que todas las familias de indicadores enunciadas en el capítulo de Estado de la Cuestión (ver Sección 2.1.2) estén, al menos, representadas. Para implementar esta etapa se ha utilizado una librería de software libre llamada

[1] Si efectuaramos el estudio solo con las empresas que han formado siempre el FTSE100, estaríamos sesgando la realidad ya que estaríamos indirectamente seleccionando empresas que han tenido éxito en dicho período.

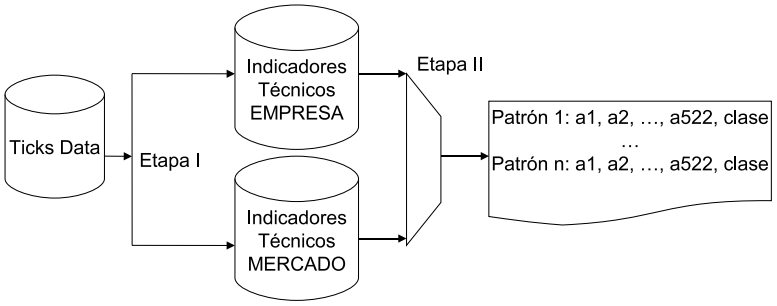

Figura 4.14: Flujo general de generación de patrones

TA-LIB [1]. Muchos de los indicadores implementados necesitan de un parámetro que especifica el número de días sobre los que se aplica. En esos casos, el indicador técnico se ha parametrizado para período con valores: 5, 10, 20, 60 y 120. De este modo, se obtienen 5 atributos por cada indicador. En la Tabla A.2 (ANEXO A) se especifican los indicadores técnicos que se han generado haciendo uso de TA-LIB (una descripción más detallada de los mismos puede consultarse en *http://www.ta-lib.org/function.html*) y los atributos a los que han dado lugar.

Además, en algunos casos particulares (RSI, ADX, WILIAMS y medias móviles), se ha calculado también el mismo indicador para el día anterior para poder efectuar más adelante cálculos sobre la variación del indicador de un día a otro y los cruces entre indicadores.

En la Tabla 4.11 se muestran los indicadores técnicos generados, para cada empresa, en esta fase I. La columna 'indicadores' indica el número de éstos que han sido generados para cada categoría. 'Períodos' indica para cuantos períodos distintos se han generado dichos indicadores. Entre paréntesis se indica el valor del parámetro período de cada indicador. Ejmplo, RSI(14), sería el indicador técnico RSI calculado para los 14 días previos. La columna 'Día anterior' indica si se ha generado también información para el día D-1. Por último, la columna 'Total' indica el total de atributos al que da lugar cada categoría de indicadores.

Además de los indicadores calculados para cada empresa y día, se han calculado indicadores para el global del mercado. Por ejemplo, Contador(+,-) cuenta la frecuencia de subidas (empresas cuya cotización ha subido frente al total de empresas cotizadas)

[1] http://ta-lib.org

para un período dado. En la Tabla 4.12 se muestran los indicadores técnicos globales generados. El significado de cada columna es similar al detallado para la tabla anterior.

Tabla 4.11: Indicadores técnicos generados en etapa I para cada empresa

Tipo	Indicadores	Períodos	Día anterior	Total
Indicadores Técnicos TA-LIB	80	5 (5, 10, 20, 60 y 120)	No	$80 \times 5 = 400$
RSI	1	3 (13, 14 y 15)	Sí	$1 \times 3 \times 2 = 6$
(ADX, WILIAMS)	2	3 (8, 10 y 12)	Sí	$2 \times 3 \times 2 = 12$
Medias Móviles	1	5 (5, 10, 20, 60 y 120)	Sí	$1 \times 5 \times 2 = 10$
Contador(+,-)	1	5 (5, 10, 20, 60 y 120)	No	$1 \times 5 = 5$
(promedios close, high, low)	3	7 (1, 2, 5, 10, 20, 60 y 120)	No	$3 \times 7 = 21$

Tabla 4.12: Indicadores técnicos globales generados en etapa I

Tipo	Indicadores	Períodos	Día anterior	Total
Medias Móviles	1	5 (5, 10, 20, 60 y 120)	Sí	$1 \times 5 \times 2 = 10$
Contador(+,-)	1	5 (5, 10, 20, 60 y 120)	No	$1 \times 5 = 5$
(promedios close, high, low)	3	7 (1, 2, 5, 10, 20, 60 y 120)	No	$3 \times 7 = 21$

En una **segunda etapa**, se procede a transformar los indicadores técnicos obtenidos en la etapa anterior, a una segunda representación numérica.

Una de las operaciones más habituales en este campo consiste en estudiar el cruce de dos líneas que suelen representar indicadores o promedios móviles. En la figura 4.15 vemos uno de estos cruces entre las líneas $Linea_1$ y $Linea_2$. Podemos pensar, a modo de ejemplo, que $Linea_1$ es el promedio móvil de los últimos 60 días y $Linea_2$ el promedio móvil de los últimos 10 días. Como podemos observar en la figura, la línea $Linea_2$ se mueve por debajo de la línea $Linea_1$ hasta el punto p. En dicho punto ambas líneas se cruzan y a partir de ahí la línea $Linea_2$ toma valores por encima de $Linea_1$. Partiendo de dos puntos de cada línea, se puede representar el valor de un indicador para dos instantes de tiempo, codificando el cruce como dos números reales: $(\frac{a_1}{a_2}, \frac{b_1}{b_2})$. Si en un instante dado, el primer número es menor a la unidad y el segundo mayor, se puede deducir que se ha producido un cruce entre ambas líneas. El valor concreto de cada cociente indica el ángulo de las líneas entre sí.

Las reglas empleadas para completar las transformaciones en esta segunda han sido:

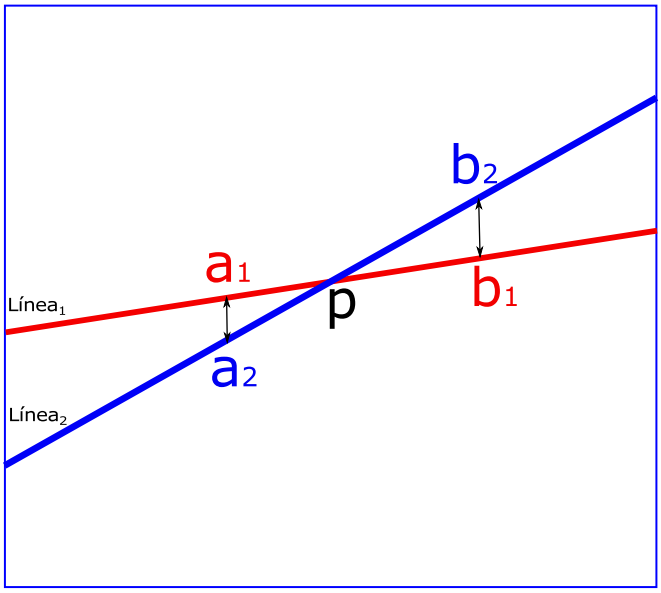

Figura 4.15: Cruce de dos líneas o indicadores

- Los Indicadores técnicos TA-LIB se han mantenido en esta segunda etapa tal cual han sido calculados en la primera.

- Para el caso particular de los indicadores RSI, ADX y WILIAMS también se ha codificado la canalización de éstos entre dos niveles o umbrales, tal como se muestra en la Figura 4.16. La idea es detectar cuando el indicador bajo observación cruza alguno de los niveles. Para detectar estas situaciones se observa el indicador en dos instantes de tiempo y se computa si para un instante $D-1$ se encuentra a un lado(arriba o abajo) de un nivel y en un instante D se encuentra al otro. Para codificar la rotura de los niveles se ha usado un método similar al presentado para el cruce de dos líneas. De esta forma, para codificar el cruce del indicador con un nivel dado, se han generado dos números reales que comparan el valor del indicador para dos instantes de tiempo con el valor de un nivel prefijado. En la Tabla 4.13 se muestran los valores concretos que se han usado para establecer el canal para cada uno de los tres indicadores.

- Las medias móviles se han transformado en una matriz de números que representan los cruces del valor de cada media con las demás. El número total de atributos necesarios para representar los cruces es: $\binom{5}{2} \times 2 = 20$.

Tabla 4.13: Canalización de RSI, ADX y WILIAMS

Indicador	Periodo(días)	Nivel inferior	Nivel superior	Total
RSI	13, 14 y 15	20	70	$3 \times 4 = 12$
ADX	8, 10 y 12	0.2	0.5	$3 \times 4 = 12$
WILIAMS	8, 10 y 12	-79	-20	$3 \times 4 = 12$

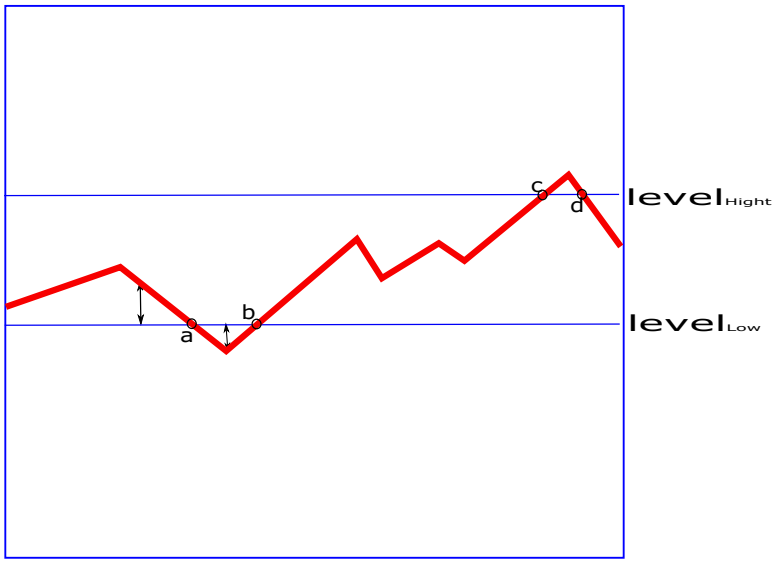

Figura 4.16: Canalización

- Los contadores pasan a esta segunda etapa tal cual.
- Promedio de *close*, *high* y *low*: se transforman en una matriz de números que comparan porcentualmente cada campo para el período de un día, con el campo equivalente de todos los demás períodos. El número total de atributos necesarios será: $6 \times 3 = 18$.

Tal como se puede observar en la Tabla 4.14 cada patrón de datos se codifica en 522 atributos, todos ellos números reales, con dos dígitos decimales.

La clase de salida (evento a predecir) de cada patrón de datos toma dos posibles valores: C_0 y C_1. Si la cotización para una empresa sube entre la tarde (minuto 480) del día D y la media mañana (minuto 120) del día D+1, el patrón es clasificado como C_1 y, como C_0 en caso contrario.

Si llamamos E a una empresa y D a un día cotizado, definimos *clase* según la Ecuación 4.2.2.

Tabla 4.14: Codificación final de cada patrón de datos

Tipo Indicador	Número de atributos
Indicadores Técnicos TA-LIB	400
(RSI, ADX y WILIAMS)	36
Medias Móviles	20
Contador(+,-)	5
(promedios close, high, low)	18
Global Medias Móviles	20
Global Contador(+,-)	5
Global (promedios close, high, low)	18
TOTAL	522

$$clase(E, D) = \begin{cases} C1 & \text{si } Precio(E, D+1, 120) > Precio(E, D, 480) \\ C0 & \text{en caso contrario.} \end{cases}$$

4.2.3. Marco experimental

Los experimentos se han dividido en dos bloques llamados Bolsa-I y Bolsa-II.

El objetivo del experimento **Bolsa-I** es evaluar la capacidad de LIA/CLIA y de Chi-Squared/Random Forest para localizar y aprovechar los atributos que pudieran aportar información para la clasificación correcta de cada patrón de datos, escondidos éstos entre el total de atributos de cada problema.

En este experimento se ha añadido un atributo extra a los datos de entrada. Dicho atributo se ha generado artificialmente de forma que contiene información relevante para predecir la clase de salida. Se han realizado tres lanzamientos experimentales, variando, en cada ocasión, la información aportada por el atributo artificial. Específicamente, se ha experimentado con atributos que permiten obtener una precisión máxima en la predicción de la clase C_1 del 55 %, 60 % y 65 %. A los escenarios que incluyen dicho atributo artificial se les ha llamado: Info55, Info60 e Info65.

En **Bolsa-II** se ha experimentado exclusivamente con datos reales, con la finalidad de dilucidar si en éstos existe información y cual de las técnicas de aprendizaje automático utilizadas en el experimento ha sido más eficaz sacando provecho de ello.

En ambos experimentos, como medida utilizada para comprobar el grado de éxito de cada técnica, se ha utilizado la *precision* en la clase C_1 para cada día D.

$PrecisionC_1(D)$ se define en la Ecuación 4.2, siendo TP el número de patrones correctamente clasificados como C_1 y FP el de patrones incorrectamente clasificados como C_1. Como resultado de cada experimento se ha obtenido una serie de datos para cada una de las técnicas de clasificación utilizadas. El valor que toma cada punto de la serie corresponde a la *precision* diaria en clase C_1, obtenida con un clasificador dado.

$$precisionC1(D) = \frac{TP}{FP + TP} \tag{4.2}$$

Parametrización:

Para este experimento LIA se ha parametrizado con: $dept = 1$ (para máxima velocidad, dado el elevado número de problemas a procesar), $maxAtrs = 8$ (para la fase II de CLIA no es conveniente, por tiempo de proceso, partir de una selección de atributos mucho mayor) y $mode = 1$ (intenta localizar los atributos que aportan zonas de máxima densidad de patrones de clase C_1). CLIA se ha configurado con $confianza = 5$ (los experimentos en sintéticos parecen confirmar que 5 es un valor adecuado para este tipo de problemas. No obstante, si bien para un problema concreto los resultados pueden variar al usar distintos valores para este parámetro, en promedio, los resultados con CLIA son muy poco sensibles para valores de confianza en el rango [3..7], por lo que el uso de 5, como valor medio, parece una elección razonable).

Como técnicas frente a las que comparar los resultados, se ha utilizado Chi-Squared, como técnica de selección de atributos, parametrizada con $maxAtrs = 8$ y, Random Forest, con parámetro $n\_estimators = 50$.

Entrenamiento:

El clasificador CLIA genera dos posibles salidas: 1 ó 0. La primera indica que la clase predicha es C_1. La segunda indica estado *Indeterminado*, es decir, indica que no existen evidencias fiables de que el patrón sea de clase C_1.

La implementación de Random Forest utilizada no permite optimizar directamente *precision*, ni contempla el estado indeterminado. Sin embargo, el clasificador sí proporciona para cada predición la probabilidad asociada. Aprovechando dicha información, se pueden optimizar los resultados de forma que se emule la misma semántica que CLIA. Así, si Random Forest clasifica un patrón como de clase 1 y la probabilidad de acierto estimada supera cierto umbral entonces, la salida final será 1. En caso contrario, si la salida es 0 ó si siendo 1 no se alcanza cierto umbral de probabilidad, entonces, la salida final será 0.

Método para optimizar *precision* y contemplar estado *indeterminado* con Random Forest:

1. Dividir el *conjunto de entrenamiento original* en *conjunto de entrenamiento* (80 % patrones) y *conjunto de validacion* (20 % patrones).
2. Entrenar la técnica con el *conjunto de entrenamiento*.
3. Predecir el *conjunto de validacion*, obteniendo la probabilidad de cada predicción.
4. Utilizando las probabilidades y la curva *precision/recall* sobre el *conjunto de validación* encontrar el umbral de probabilidad que maximize *precision* para $recall >= 10\,\%$.
5. Utilizando el modelo entrenado y el umbral de probabilidad, predecir el *conjunto de test*.

Una alternativa que puede utilizarse para hacer más robusto el sistema, podría ser utilizar validación cruzada en el *conjunto de entrenamiento* , de forma que repitieramos la estimación del umbral de probabilidad tantas veces como $folds$ utilizáramos. El inconveniente que puede tener esta alternativa es el mayor tiempo de proceso necesario para procesar cada uno de los $folds$. Por ello, en este experimento se ha utilizado un único *conjunto de validación*.

4.2.3.1. Metodología

Partiendo de la hipótesis que planteamos al principio de esta tesis: *el mercado es dinámico y la información utilizable aparece y desaparece rápidamente* se ha diseñado el experimento con la idea de que el sistema aprenda de un conjunto de datos, cercanos en el tiempo, utilice la información extraída, brevemente, y repita el ciclo de aprendizaje con nuevos datos.

Para ello, la metodología ha consistido en entrenar, para cada empresa cotizada, la técnica de aprendizaje automático con patrones de datos de una ventana histórica relativamente corta (300 días) y, una vez obtenido el modelo correspondiente se ha usado éste para realizar predicciones durante el tiempo más breve posible (1 día). El proceso se ha repetido para todos los instantes de tiempo considerados, 1.934 días y para las 173 empresas que alguna vez han conformado el FTSE100. De esta forma,

se pretende que el modelo utizado para predecir cada día, se base en los datos más recientes disponibles de forma que su utilidad no empieze a decaer por obsolescencia.

Dado que para predecir un día D necesitamos los 300 días anteriores (para entrenar y deducir un modelo) y siendo que alguno de los atributos de cada día computa u observa hasta 120 días anteriores, el primer día que puede ser predicho por el sistema es $D_{inicio} = 300 + 120 = 420$.

El número de problemas que forman el experimento puede calcularse como *NProblemas=(empresas cada día) x (días cotizados válidos)*. Siendo un día cotizado válido para la empresa E, aquel para el cual se cumplen dos condiciones: 1. La empresa E ha cotizado dicho día como parte del FTSE100 y, 2. Disponemos de 420 días previos para entrenar el modelo. El máximo de problemas sería así de $100 \times (1934 - 420) = 151\,400$. En la práctica, dado que las empresas entran y salen del índice FTSE100 varias veces, a lo largo del experimento, el número total de problemas válidos ha sido de **117 207 problemas**.

La operativa que se haría en el mercado a partir de dicho modelo consistiría en adquirir cada día D las empresas clasificadas como de C_1, y venderlas al día siguiente. Es decir, se usaría la técnica de clasificación para realizar una selección diaria de empresas en las que invertir. Si la clasificación como C_1 es correcta, en la operación de compra/venta habremos ganado dinero. Si no lo es, habremos perdido. Dado que solo operamos para empresas clasificadas como C_1, los patrones clasificados como C_0 no generan ni pérdidas ni ganancias. Por ello, y desde una perpectiva de aprendizaje automático, la medida que vamos a intentar optimizar es $precision = (TP/TP + FP)$.

4.2.3.2. Línea base con la que comparar

Los experimentos se han valorado desde dos puntos de vista: un primer enfoque donde se analiza si la frecuencia de aciertos utilizando la metodología presentada supera a la esperable por azar y, un segundo enfoque, donde se cuantifica la posible utilidad económica del uso de la técnica frente a estrategias pasivas o semipasivas de compra venta de valores.

Como punto de partida, se ha computado la frecuencia trivial de la clase $C1$ (ver Ec. 4.3) para el total de empresas y período considerado. El valor obtenido ha sido de 0.47. Así, cualquier subconjunto de operaciones de compra/venta que seleccionemos

tenderá a acertar un 47 % de las ocasiones, salvo que la técnica en cuestión encuentre información útil en los datos.

$$frecuenciaTrivialC_1 = \frac{|C_1|}{|C_0| + |C_1|} = \frac{55\,087}{117\,207} \approx 47\,\% \tag{4.3}$$

En la sección de resultados se comparará la frecuencia de aciertos de cada técnica con este 47 %. Además, también se comparará la frecuencia de aciertos para cada empresa por separado. Por último, y de cara a comprobar si los resultados son robustos a lo largo del tiempo, se van a comparar los aciertos en cada día D que dura el experimento.

En la Ecuación 4.4 se define $frecuenciaTrivialDiariaC_1(D)$, la frecuencia trivial para un día D. Se calcula como el número de empresas que deben ser clasificadas como C_1, el día D, respecto al total de empresas cotizadas dicho día. Dado que en el FTSE100, en general, diariamente cotizan 100 empresas, el denominador de dicha fórmula suele tomar el valor 100.

Puesto que la clase C_1 corresponde a patrones donde la diferencia de cotizaciones de acciones de una empresa, entre dos instantes de tiempo, es positiva podemos interpretar también dicha función como el promedio de empresas que suben en bolsa el día D respecto al total de empresas cotizadas ese día.

$$frecuenciaTrivialDiariaC_1(D) = \frac{|C_1(D)|}{|C_0(D)| + |C_1(D)|} \tag{4.4}$$

Para ello, se ha obtenido la serie Market presentada en la Figura 4.17. Cada valor de la serie Market (ver Ec. 4.5), para un día D, se calcula con el promedio de los últimos 100 días de $frecuenciaTrivialDiariaC_1(D)$. Con $D \in [420..1934]$.

$$Market[D] = \frac{1}{100} \times \sum_{n=D-99}^{D} frecuenciaTrivialDiariaC_1(D) \tag{4.5}$$

El motivo de la presentación del promedio móvil de 100 días, es suavizar la curva de forma que se aprecie con claridad la tendencia a lo largo de todo el experimento. La bajada que se observa alrededor de los puntos 600-800 corresponde al impacto que tuvo la crisis en el mercado de valores en el año 2008.

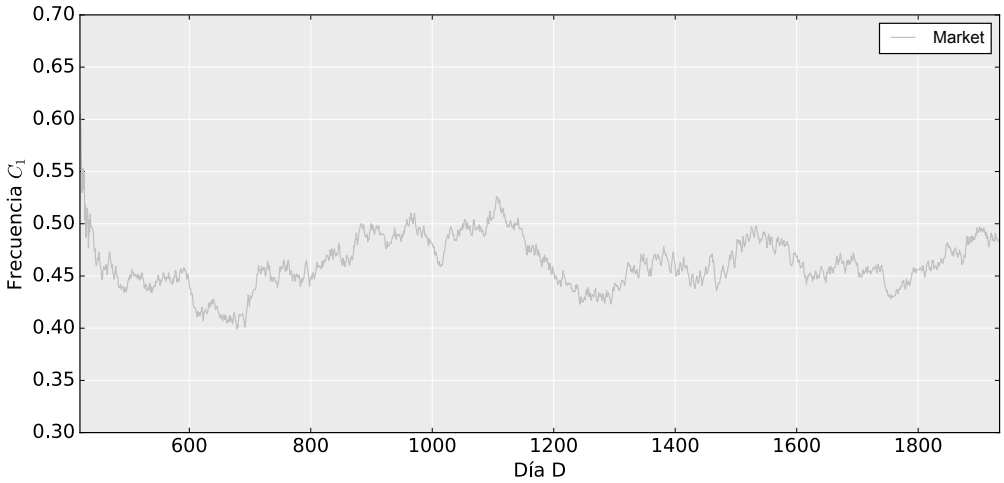

Figura 4.17: Promedio móvil de $frecuenciaTrivialDiariaC_1$

Al objeto de valorar el rendimiento económico de la nueva técnica, se ha precalculado el rendimiento que obtendríamos utilizando una estrategia totalmente pasiva. La estrategia *Comprar y Mantener* consiste en comprar acciones al inicio del experimento y venderlas a la finalización de éste. Se denomina pasiva porque no requiere atención, ni realizar operaciones durante el transcurso de la inversión. Esta característica tiene, entre otras ventajas, el ahorro en comisiones de compra/venta y un mejor trato fiscal, ya que las posibles ganancias solo tributan cuando se realizan (cuando se venden las acciones) por lo que la estrategia pasiva se dice que difiere los impuestos hasta la venta de los activos. Dentro de las posibles variantes a esta técnica, en este trabajo se ha usado el rebalanceo automático de valores dentro de la cartera. Ésto es, el sistema automáticamente ha balanceado el peso económico de las acciones de cada empresa en la cartera, de forma que cada una suponga exactamente el 1 % (el FTSE100 lo componen cada día 100 empresas) del capital total en un momento D. A efectos del cálculo de rentabilidad se ha decidido no cargar costes de transación a dicha operativa de rebalanceo ya que es común utilizar productos financieros llamados "Fondos indexados"que realizan este rebalanceo diario/períodico sin cargar comisiones de compra/venta al inversionista. Sí se ha imputado un coste equivalente al 1 %/anual del capital. Dicho coste correspondería a la comisión de custodia de acciones (1 % anual) o a los gastos de gestión de un fondo indexado, en caso de que se usara dicha herramienta de inversión. Para ambos casos, se

ha considerado razonable cargar el 1 % al año. El porcentaje de capital invertido (*capital in-Market*) respecto al capital total es, en esta estrategia del 100 % (todo el dinero lo tenemos invertido en acciones todo el tiempo). La rentabilidad obtenida utilizando la estrategia *Comprar y Mantener*, para el total de empresas y período considerado, ha sido del 30 % (Aproximadamente 3,75 % TAE). Dicho cálculo de rentabilidad no ha tenido en cuenta el cobro de dividendos durante el experimento.

Las siguientes medidas, Ecuaciones 4.6 y 4.7, pueden usarse para evaluar sistemas en el ámbito de la compra/venta de acciones en el mercado de valores Malkiel (1973a). Ámbas se basan en comparar la rentabilidad obtenida al utilizar el sistema a evaluar frente a otra estrategia trivial. Es necesario parametrizar las medidas aportando, como constantes, la comisión de compra venta de acciones, $comisionCV$, y, la utilidad económica de acertar o fallar un patrón clasificado como de clase $C1$, $UtilTP$ y $UtilFP$ respectivamente.

- $Renta_0$: rentabilidad frente estrategia *Comprar y Mantener*.

$$Rentabilidad_0 = \frac{TP * Util_{TP} + FP * Util_{FP}}{precio_{fin} - precio_{ini} - 2 * comisionCV} \tag{4.6}$$

- $Renta_1$: rentabilidad frente estrategias de compra y venta. Donde N es el total de operaciones realizadas.

$$Rentabilidad_1 = \frac{TP * Util_{TP} + FP * Util_{FP}}{\sum_{i=1}^{N}(precio_i - precio_{i-1} - 2 * comisionCV)} \tag{4.7}$$

Una debilidad que presenta la estrategia de *Comprar y Mantener*, consiste en que es muy afectada por el momento inicial y final elegidos por el experimento. Para ofrecer una estimación quizás más realista de una estrategia pasiva, se ha calculado también lo que se ha llamado *Compra Continua a Inversión Constante*. Consiste en considerar que haciendo uso de una gestión del capital adecuada, invertimos cada día X unidades monetarias. X supondrá un % de nuestro inicial total, quedando el resto en reserva. Para este experimento, se ha establecido que la inversión diaria se realiza por un máximo del 50 % del capital inicial. $Inversiondiaria(diaD) = Min(capitalInicial \times 0.50, capital(diaD))$. Cada día se redistribuye el capital disponible de forma que el capital *in-Market*, en un día dado, no supere el capital de inversión para dicho día y que este se reparta equitativamente entre todas las empresas. El procedimiento para reequilibrar la cartera de acciones, puede hacerse por medio de

compra/venta de acciones o por medio de un fondo indexado, tal como se apuntó anteriormente para el caso de la estrategia *Comprar y Mantener*. Al mantener la inversión constante a lo largo de todo el experimento, esta segunda estrategia es menos variable a los momentos elegidos de inicio y fin. La rentabilidad obtenida utilizando esta estrategia, más conservadora, aplicando los mismos gastos y comisiones, ha sido del 45 % (Aproximadamente 5,6 % TAE).

Por último, se ha calculado la rentabilidad utilizando una estrategia que podríamos llamar *Dummy*. Consiste en comprar todos los días, todas las empresas, a la hora elegida (minuto 480) y vender todas al día siguiente en el minuto 120. Es decir, replicar la operativa que pretendemos hacer utilizando las técnicas de aprendizaje automático pero comprando y vendiendo el total del mercado. Dado que esta última estrategia requiere un número muy alto de operaciones el resultado final está muy influenciado por las comisiones concretas a aplicar a cáda operación. Para implementar la operativa real podemos pensar en compra/venta de acciones, pero también podríamos utilizar otros recursos: futuros sobre acciones, opciones, ETFs y toda una variedad de productos y brokers. El porcentaje de capital invertido respecto al capital total se ha establecido en esta estrategia en el 50 % (la mitad[1] del dinero lo tenemos invertido en acciones todo el tiempo). Con el fin de ofrecer unas cifras realistas, en la Tabla 4.15 se muestran los rendimientos para tres escenarios distintos de comisiones.

Tabla 4.15: Rentabilidad de la estrategia *Dummy*

Comisiones de compra/venta	Rentabilidad estrategia *Dummy*
0,0025 %	20 % (Aproximadamente 2,5 % TAE)
0,025 %	1 %
0,125 %	-52 %

En la Figura 4.18 se muestran tres series de rentabilidad, generadas por inversión pasiva, para todo el período experimental. La estrategia pasiva más rentable en este período ha sido *Compras a Inversión Constante* que ha obtenido casi un 45 % de rentabilidad. La segunda serie más rentable ha sido *Comprar y Mantener*, que ha obtenido

[1] Es habitual en la literatura aplicar porcentajes conservadores de dinero invertido. Habitualmente, del 3 %. Es decir, en un momento dado, solo arriesgamos el 3 % de nuestro dinero. Dado que la estrategia aquí planteada utiliza acciones y no derivados más riesgosos o apalancados y dado que el mercado de acciones raramente fluctua en su conjunto más del 5 % diario, tener invertido el 50 % de nuestro capital estimamos supone el riesgo de perder en una jornada un, asumible, 2.5 % de nuestro capital total.

un 30 % en el mismo período. Por último, la serie menos rentable ha sido *Dummy*, para Comisión1 (0,0025 %). *Dummy* ha conseguido un 20 % de rentabilidad. Por otro lado, *Dummy* ha sido la estrategia menos afectada por el período de crisis del año 2008. En dicho punto, *DummyC*1 ha sido la estrategia que más capital inicial ha mantenido.

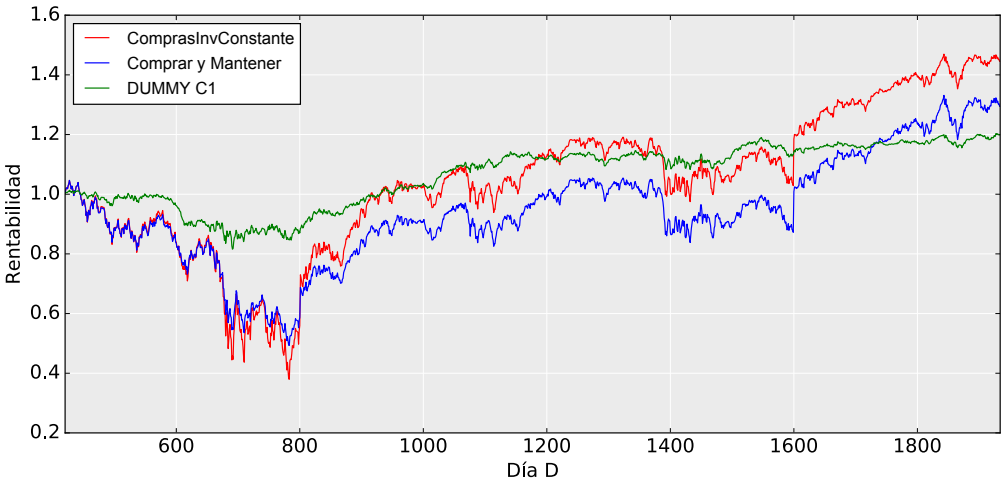

Figura 4.18: Comparación rentabilidad estrategias pasivas

4.2.4. Resultados experimentales en Bolsa-I

En la Figura 4.19 se presentan los resultados de CLIA y Random Forest, para cada uno de los escenarios artificiales planteados. La figura está dividida en tres secciones horizontales que corresponden a Info55, Info60 e Info65.

La serie Market representa el comportamiento global del mercado, frente al que comparar el resultado de aplicar una técnica de aprendizaje. El procedimiento para calcular esta serie se describe en la Sección 4.2.3.2.

Las series CLIA y Random Forest se obtienen siguiendo el procedimiento descrito en la Sección 4.2.3.1. Para mejorar la claridad de visualización y poder percibir mejor tendencias a medio plazo, la figura representa, para cada punto de las series de datos, el promedio de una ventana móvil de los 100 días previos a cada punto.

Como podemos observar, los resultados para CLIA, línea roja, generalmente, son superiores a Market. Esto es así para los tres escenarios (Info55, Info60 e Info65). Lógicamente, a mayor información en los datos, mejor desempeño del clasificador. Así,

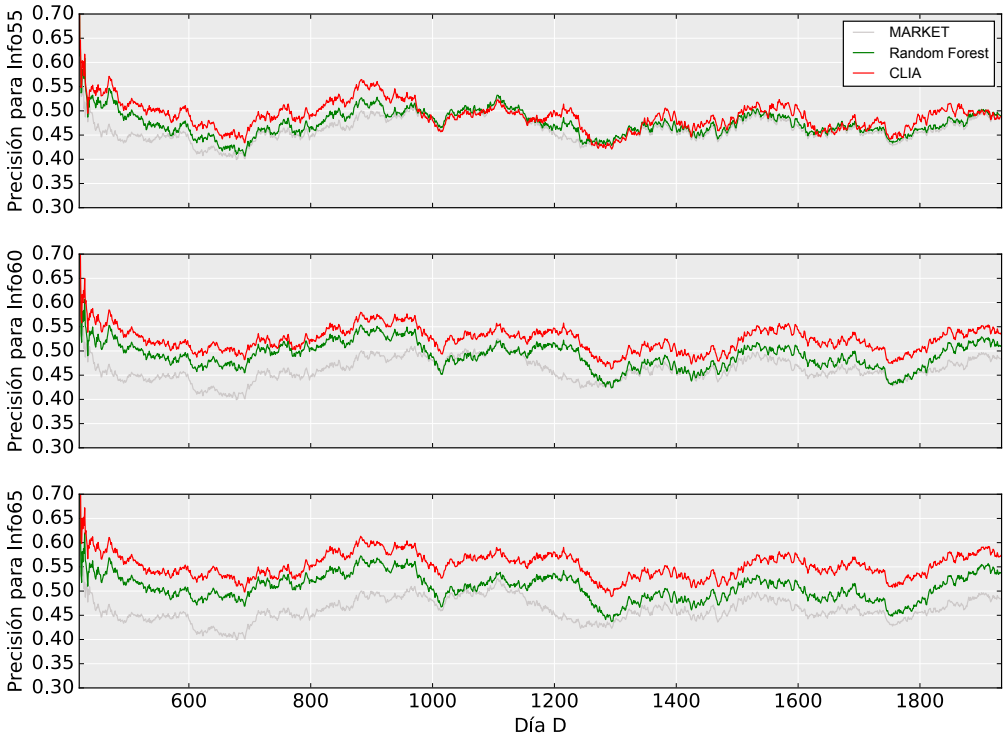

Figura 4.19: Resultados en BOLSA-I

la diferencia, punto a punto, entre la series CLIA y Market es menor para Info55 y se incrementa para Info60 e Info65.

En el caso de Random Forest, la línea verde, para el escenario Info55, está muy cercana a Market. Por ello, se considera que Random Forest no está siendo capaz de encontrar prácticamente nada de información en este escenario. Para el caso de Info60 e Info65, los resultados mejoran y, se puede observar como esta técnica también es capaz de obtener difencias positivas respecto a Market, para la casi totalidad de días que abarca el experimento.

En cuanto a la comparación entre ambas técnicas, puede observarse como, en los tres escenarios, CLIA domina casi en su totalidad a la Random Forest.

En cualquier caso, es importante tener en cuenta que para la realización del experimento se ha añadido información artificial, pero los modelos aprendidos por los clasificadores han tenido acceso no solo a los atributos artificiales si no al total de atributos ($522 + 1 = 523$). Por ello, cada escenario (Info55, Info60 e Info65) contendrá

los resultados obtenidos a partir de la información artificial y la que de forma natural existiera en los 522 primeros atributos. Así, si la información presente en los atributos reales, fuera superior a la aportada por el atributo artificial, obtendríamos unos valores de la serie en cuestión, superiores a los obtenidos únicamente del atributo artificial, ya que los clasificadores sacarán partido de los atributos que consideren más informados, no teniendo necesariamente que utilizar los atributos artificales.

Dado que $precision(Info55) < precision(Info60) < precision(Info65)$, podemos afirmar que, en caso de existir información en los atributos reales, ésta será menor a la equivalente a Info60. Ya que, de ser superior, la *precision* alcanzada en el escenario Info55, aprovechando esos atributos naturales, sería similar a Info60.

Como conclusiones finales de Bolsa-I, podemos deducir que: 1. CLIA será capaz de aprovechar eficazmente atributos que aporten como mínimo un 55 % de *precision*. En el caso de Random Forest, es probable que sea necesario contar con atributos que aporten como mínimo un 60 % de *precision* para que ésta pueda aprovecharlos eficazmente. Y, 2. La cota superior de la información (diferencia entre la *precision* y la $frecuenciaTrivial$) presente en los atributos reales se puede establecer, *a grosso modo*, en $60\,\% - 47\,\% = 13\,\%$.

4.2.5. Resultados experimentales en Bolsa-II

En este experimento, utilizando exclusivamente datos reales de bolsa, se ha intentado dilucidar si en éstos existe información aprovechable y cual de las técnicas de aprendizaje automático utilizadas (CLIA y Random Forest) es más eficaz sacando provecho de ella. La metodología y las medidas de rendimiento utilizadas, son equivalentes a las descritas para Bolsa-I.

En la Figura 4.20 se muestran los resultados obtenidos con CLIA y Random Forest para cada día del experimento. Cada punto de la serie corresponde al promedio móvil de los últimos 100 días de la *precision* en la clase C_1, obtenida con cada técnica de clasificación

A efectos de línea base con la que comparar, se muestra también la serie Market, que representa el comportamiento global del mercado. El procedimiento para calcular esta serie se describe en la Sección 4.2.3.2.

Como puede observase, CLIA, la curva roja, domina prácticamente siempre a las otras dos curvas. Es de destacar, también, el excelente comportamiento hasta el día

1000. Dicho instante de tiempo corresponde al 15 de enero del 2010. Se desconoce el por qué de este efecto.

Random Forest muestra un resultado intermedio entre CLIA y Market. Así, siempre obtiene una *precision* menor que CLIA siendo, en todo caso, para la primera mitad de los días del experimento superior a Market.

En la Figura 4.21 se ha representado la diferencia que se obtiene al usar CLIA y Random Forest sobre la serie Market. Valores positivos corresponden a ganancias de información. Hasta el instante 1000, ambas técnicas han obtenido diferencias positivas. CLIA siempre se ha mantenido por encima de Random Forest. A partir de dicha fecha, se ha producido un repunte importante el día 1200.

El resto de los días, CLIA se ha mantenido por encima del 0 % pero no ha llegado a obtener diferencias mayores al 5 %. RandomForest, por su parte, ha obtenido diferencias cercanas al 2.5 % y, muchos días, las diferencias con Market han sido negativas.

Con ambos clasificadores, existen ocasiones en las que el resultado obtenido al usar el clasificador es peor que el propio mercado, es decir, existen ocasiones donde las diferencias frente a Market son negativas. Una posible explicación a este efecto podría ser la existencia de cambios de tendencia del mercado que hagan que el comportamiento de éste cambie bruscamente invalidando el modelo aprendido.

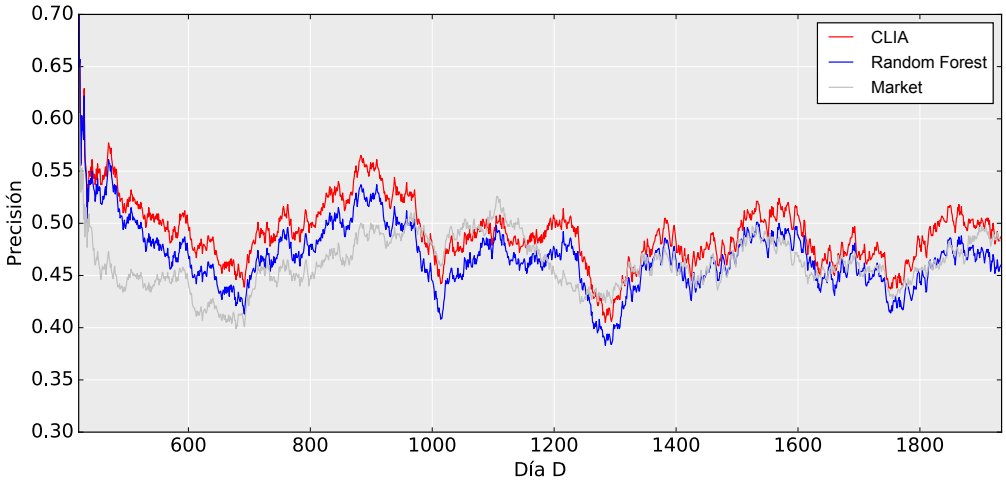

Figura 4.20: Resultados en BOLSA-II: CLIA vs Random Forest

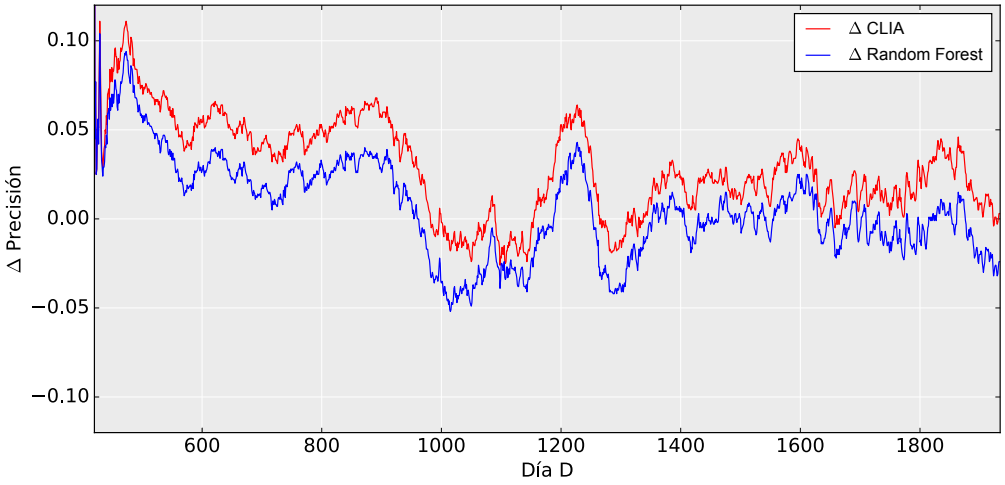

Figura 4.21: Resultados BOLSA-II - Δ CLIA *vs* Δ Random Forest respecto a Market

En la Figura 4.22 se compara el resultado de CLIA, línea roja, aplicado solo a datos reales del mercado, frente a los obtenidos con CLIA utilizando los escenarios Info55 e Info66, donde se añaden atributos generados artificialmente.

Al igual que como ya se ha explicado en la discusión de los resultados para Bolsa-I, hay que tener en cuenta que la información que contiene el escenario Info55 es, además de la del atributo añadido artificialmente y que proporciona un 55 % de *precision*, la que estuviera presente de forma natural en el conjunto de atributos reales.

Por ello, el que la serie de resultados CLIA esté muy cercana, por abajo, a la de Info55 indica que la información presente en los 522 atributos equivale aproximadamente a la que proporcionaría un atributo que permita alcanzar una *precision* del 55 %.

En la Tabla A.5 (ANEXO A) se muestra, para cada empresa, el número de días cotizados válidos, la frecuencia de C_1 para total de días cotizados, el número de días operados haciendo uso de CLIA y el promedio de *precision* para la clase C_1 los días operados.

Como puede observarse, para el total de empresas, se cuenta con 117 207 días cotizados válidos. La frecuencia de C_1 es del 47 %. Y, utilizando CLIA, se han realizado 8742 operaciones, alcanzándose un promedio de *precision* en la clase C_1 del 52 %.

Con el fin de verificar si la *precision* alcanzada con CLIA es conseguida gracias a unos pocos atributos o, si por el contrario, la información está repartida uniformemente entre todos los atributos, se ha generado la Figura 4.23. La *precision* de cada

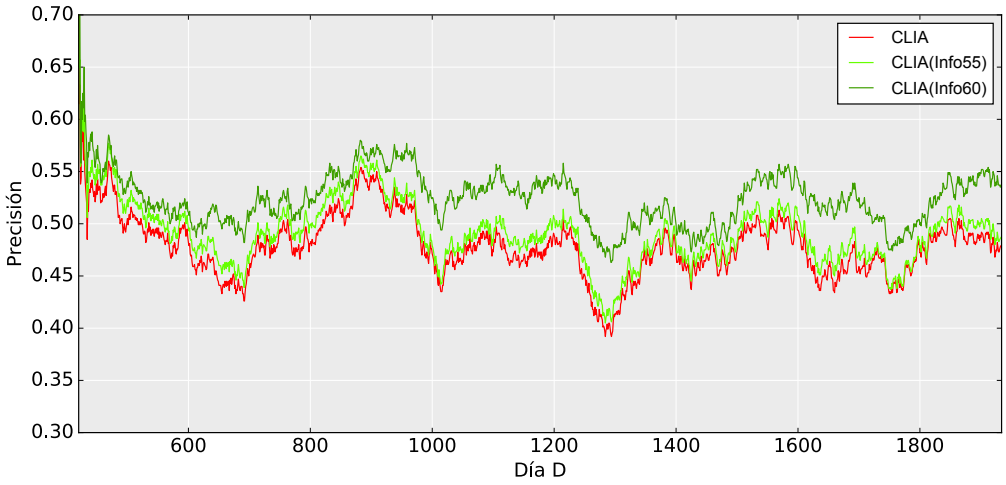

Figura 4.22: Resultados BOLSA-II - CLIA*vs* CLIA(Info55 e Info60)

atributo se ha calculado haciendo el promedio de aciertos en la clase C_1, para todas las ocasiones donde se ha usado una regla que incluya el atributo en cuestión. La serie $PrecisionFamilia$ muestra la misma información pero agrupada por familias de atributos. La serie $Trivial$, línea roja, muestra el valor de la $frecuenciaTrivial$ (que corresponde a 0.47, para el total de días y de empresas que forman el experimento). Valores de *precision* superiores a este, podrían indicar la existencia de información.

Puede observarse como los atributos que proveen de mayor *precision*, están repartidos entre los 522 atributos. No obstante, entre los últimos 100 parece que se concentran varios atributos que proveen mayor *precision*. Dichos atributos corresponden a los indicadores sencillos que se calculan a partir del estado global del mercado. Observando la serie $PrecisionFamilia$, se puede deducir que unas pocas familias de atributos parecen aportar la mayor parte de la información.

En la Tabla 4.16 se identifican las familias de atributos donde se concentran los mejores resultados.

Por último, en la Figura 4.24 se muestra un estudio de rentabilidad. Valores superiores a la unidad indican rentabilidad positiva de la inversión. Las series representadas de color verde muestran la rentabilidad que se obtendría utilizando cada una de las tres estrategias pasivas enumeradas en la Sección 4.2.3.2: *Compras a Inversión Constante*, *Comprar y Mantener* y *Dummy*. Como podemos observar, una vez superado el período de fuerte recesión hasta el año 2010, las estrategias pasivas han obtenido

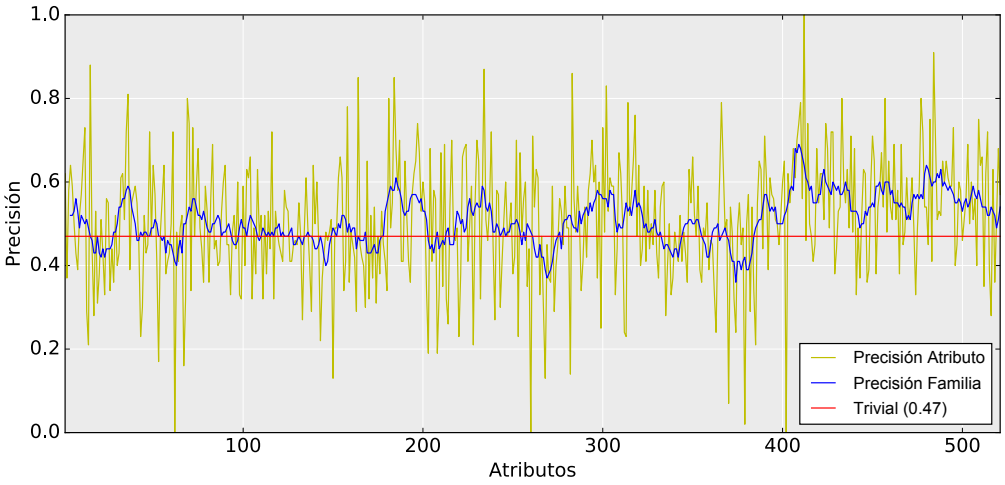

Figura 4.23: Precisión obtenida con reglas que incluyan cada atributo

Tabla 4.16: Indicadores que aportan mayor precisión

Familia Indicador	Atributo inicial	Atributo final	Promedio Familia
TA_LIB(ATR)	30	35	52 %
TA_LIB(CLDBREAKAWAY)	70	80	57 %
TA_LIB(CDLINVERTEDHAMMER,CLDKICKING)	180	200	57 %
TA_LIB(MIDPOINT,MINMAX)	290	300	52 %
RSI	400	410	57 %
WILIAMS	420	435	60 %
Contador	456	461	61 %
Global Medias Moviles	479	498	59 %
Global Contadores	499	505	56 %

ganancias. *Dummy* ha sido la estrategia más conservadora tanto en pérdidas como en ganancias.

La línea azul, representa las ganancias obtenidas utilizando Random Forest, en un escenario con comisiones C1 (0,0025 %). Por último, la línea roja representa la rentabilidad que se obtendría haciendo uso de CLIA, aplicando las mismas comisiones.

Es de destacar la superioridad de esta última técnica sobre el resto, para todo el período tratado. También resulta interesante resaltar que tanto con CLIA como con Random Forest, un inversor se habría mantenido sin soportar pérdidas en todo el período del experimento, incluida la fuerte recesión observada alrededor del punto 800. No obstante, conviene valorar con cierta prudencia los resultados de esta figura: si las comisiones aplicadas fueran C2 ó C3, la rentabilidad de las estrategias activas (CLIA y Random Forest) caerían incluso a valores negativos.

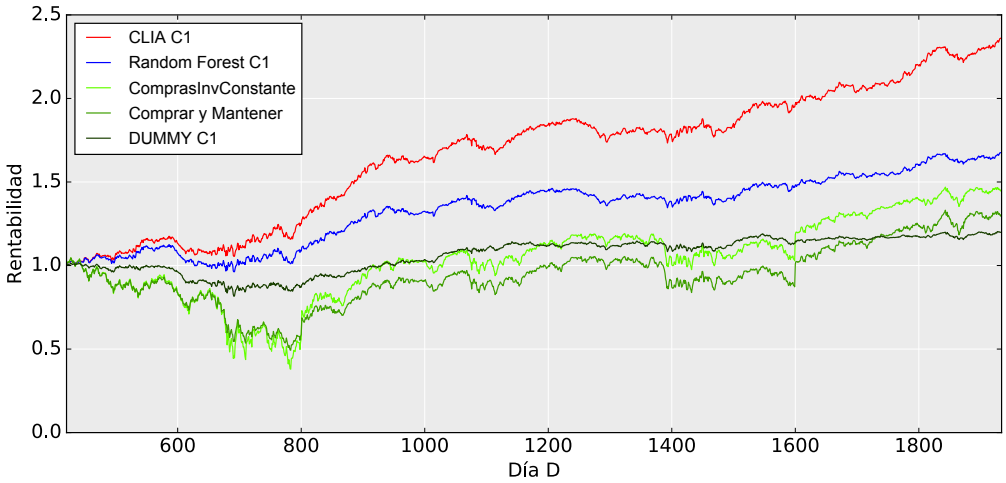

Figura 4.24: Rentabilidad de CLIA

4.3. Resumen de los resultados para el mercado de valores

En **Bolsa-I** se ha añadido un atributo extra a los datos de entrada. Dicho atributo se ha generado artificialmente de forma que contiene información relevante para predecir la clase de salida.

De este primer bloque experimental se puede concluir que ambas combinaciones de técnicas (LIA/CLIA y Chi-Squared/Random Forest) son capaces de extraer información útil de los atributos informados o relevantes presentes en el *conjunto de entrenamiento*. LIA/CLIA ha obtenido resultados superiores frente a Chi-Squared/Random Forest, siendo capaz de aprovechar mejor los atributos que aportan pequeñas ganancias de *precision* (55 %).

En un segundo bloque experimental, **Bolsa-II**, se ha experimentado con datos exclusivamente reales, sin información artificial añadida. Como puede observarse en la Figura 4.20, LIA/CLIA ha obtenido una diferencia positiva respecto a la $frecuenciaTrivialC_1$, representada en las figuras por la serie Market. Chi-Squared/Random Forest también ha obtenido diferencias positivas respecto a Market pero, para la mayor parte de los días que cubre el experimento, dicha diferencia ha estado por debajo de LIA/CLIA. Es destacable que LIA/CLIA se ha mantenido, para la práctica totalidad del experimento, en valores positivos. En el caso de Chi-Squared/Random Forest ha estado por encima de Market hasta, al menos, el día 1100 del experimento. Para ambas técnicas, puede

apreciarse que el rendimiento del sistema parece decrecer en el tiempo, obteniéndose menores diferencias de aciertos en los años más recientes.

En la Tabla A.5 se ha estudiado el resultado obtenido particularizado para cada empresa cotizada. Como puede observarse, la mejora obtenida lo ha sido para muchas empresas, no concentrándose de forma especial en ninguna.

Comparando los resultados obtenidos por cada técnica, en Bolsa-I y Bolsa-II, y observando que la $precisionC_1(D)$ con datos solo de mercado están para LIA/CLIA muy cercanos a los obtenidos en Info55, podemos deducir que: la información presente en los datos de entrada (indicadores técnicos sobre datos históricos de bolsa) parece equivaler a la del escenario Info55. Dado que la $frecuenciaTrivialC1$ es del 47 %, podemos estimar que LIA/CLIA es capaz de obtener una diferencia positiva del 55 %-47 %=8 %. De forma similar, se puede calcular que la diferencia para Chi-Squared/Random Forest es del 4 %, aproximadamente la mitad.

Se ha querido verificar, también, si existen unos pocos atributos que permitan alcanzar dicha *precision* o, si por el contrario, la información está repartida entre muchos de los atributos utilizados como entrada. Las Figuras 4.23 muestra como los atributos potencialmente informados (atributos que han dado lugar a predicciones correctas), están bastante repartidos entre el total de 522 atributos.

En la Tabla 4.16 se muestran las familias de atributos donde se concentran los mejores resultados. Puede observarse como los indicadores sencillos, que se calculan a partir del estado global del mercado, parecen concentrar mayor utilidad informativa que el resto.

En la Figura 4.24 se muestra el rendimiento económico del sistema frente a otras estrategias pasivas. Tal como era de esperar, dado el mayor porcentaje de aciertos, LIA/CLIA obtiene rentabilidades superiores al resto de técnicas, siempre y cuando se considere un escenario de bajas comisiones.

Vistos en conjunto, estos resultados parecen confirmar que la medología utilizada, la generación de los patrones y el uso de LIA/CLIA para analizarlos, permite obtener de manera estable en el tiempo, aciertos superiores a los esperables por azar. La diferencia sobre $frecuenciaTrivialC1$ se ha podido estimar, en el caso de LIA/CLIA, entre un 5 % (atendiendo a resultados por empresa) y un 8 % (a partir de comparación de resultados frente Info55).

5

Validación experimental en problemas de clasificación y regresión

En este capítulo se validan experimentalmente las nuevas técnicas de selección de atributos para problemas de clasificación y regresión. Se ha utilizado primeramente un conjunto de dominios de clasificación, obtenidos de repositorios públicos. En un segundo bloque experimental, se han validado las nuevas técnicas de selección en problemas de regresión. Dentro de este último se ha estudiado, también, el desempeño de las nuevas técnicas de selección de atributos en un problema de predicción fotovoltaica, caracterizado por presentar un gran número de atributos.

5.1. Dominios de clasificación

5.1.1. Marco experimental

En este bloque de experimentos se quieren validar las diferentes variantes de LIA sobre un conjunto de dominios de clasificación, con diferente grado de desbalanceo de clases, número de atributos y número de instancias, obtenidos de repositorios públicos.

El rendimiento de cada variante de LIA se ha comparado frente a tres técnicas, tipo *Filter*, de selección de atributos: CFS-Subset, ReliefF y Chi-Squared, incluidas en la herramienta WEKA (Witten et al., 2011). Para una descripción de cada técnica consultar Capítulo de Estado de la Cuestión (ver Sección 2.4.1).

Para las tres técnicas, todos los parámetros han tomado el valor por defecto establecido en WEKA. Como algoritmos de búsqueda asociados a CfsSubsetEval se han utilizado: ExhaustiveSearch para dominios de menos de 25 atributos y, BestFirst para el resto.

Todas las variantes de LIA han usado la parametrización por defecto a excepción de LIA que se ha configurado, para estos experimentos, haciendo uso del parámetro $mode = 2$. Este modo de funcionamiento se ha diseñado con el objetivo de ordenar los atributos según la capacidad de cada atributo de aportar información (reducción de entropía) a la solución del problema, ya sea solo o en combinación con otros.

Para evaluar la lista de atributos obtenidos por cada técnica de selección de atributos, se han utilizado éstos con diferentes clasificadores: J48, K-NN y MLP (todos incluidos en WEKA). Puede consultarse una descripción de cada técnica en el Capítulo de Estado de la Cuestión en la Sección 2.4.2.

Para evaluar los resultados de los clasificadores se ha usado *G-mean*, Ecuación 5.1. *G-mean* se define como la media geométrica que combina sensibilidad y especificidad. Resulta adecuada para evaluar la bondad de la clasificación en dominios desbalanceados ya que es independiente de la abundancia de número de patrones de cada clase.

$$G\text{-}mean = \sqrt{\underbrace{\frac{TP}{TP+FN}}_{Sensibilidad} \times \underbrace{\frac{TN}{TN+FP}}_{Especificidad}} \tag{5.1}$$

En estos experimentos se ha usado validación cruzada, con 5 $folds$ estratificados para cada dominio.

Así, para validar el rendimiento de cada técnica se ha procedido de la siguiente manera: se ha seleccionado, con cada técnica y para cada uno de los 5 $folds$ estratificados en los que se ha particionado cada dominio, un tanto por ciento de atributos (de 10 % a 90 %). Una vez obtenidos éstos, se ha entrenado un clasificador utilizando solo los atributos seleccionados.

A continuación, se ha calculado *G-mean* obtenida al aplicar el clasificador al problema de test correspondiente, filtrando por los atributos seleccionados.

A continuación, se ha calculado *G-mean* del dominio, con el promedio obtenido para los 5 $folds$.

El procedimiento se repite para cada porcentaje de atributos, en pasos de 10 % y, para cada uno de los dominios.

En las figuras se usa el Promedio Normalizado de *G-mean*. Para calcular éste se procede de la siguiente manera: para cada dominio, una vez hecha la media de los 5 $folds$, se calcula el mejor resultado obtenido (para cualquier tanto por ciento de atributos y algoritmo) y se utiliza para normalizar el resto de resultados. Así, en cada dominio, todos los resultados estarán en el rango [0..1]. A continuación, se calcula el promedio para todos los dominios, utilizando para cada uno los valores normalizados. Valores mayores de promedio de *G-mean* indican mejor desempeño.

Finalmente, se ha usado un *Test Wilcoxon Signed-Rank* (Wilcoxon, 1945; Royston, 1982; Demšar, 2006) para determinar si la diferencia de resultados entre las técnicas a comparar es estadísticamente significativa. Este test, se utiliza para comparar dos mediciones relacionadas y determinar si la diferencia entre ellas es estadísticamente significativa. Al ser una prueba no paramétrica de comparación de dos muestras relacionadas no presupone ni necesita de una distribución específica. Por ello, se utiliza en preferencia a la prueba de *t de student* cuando no se puede garantizar la normalidad de las muestras.

5.1.2. Descripción de los dominios de clasificación

Como dominios de clasificación se ha utilizado un conjunto de problemas binarios (2 clases) obtenidos de los repositorios públicos KEEL (KEEL-dataset repository)[1] (Alcalá-Fdez et al., 2011) y UCI Machine learning repository [2] (Asuncion and Newman, 2007). 15 dominios proceden de KEEL y 3 (Musk-2, Splice, and Kr-vs-kp) de UCI. Para todos los dominios se ha usado la versión particionada para validación cruzada, con 5-$folds$ estratificados.

En la Tabla 5.1 se muestra el nombre de cada dominio, el número de instancias, el número de atributos y el ratio de desbalanceo entre las clases (IR), medido como se propone en (Fernández et al., 2008), $IR = \frac{|C_1|}{|C_0|}$, siendo C_1 la clase minoritaria.

Tabla 5.1: Descripción de los dominios reales

Dominio	Nº Atributos	Nº Instancias	IR
Musk-2	166	5279	5.49
Sonar	60	166	1.16
Splice	60	2540	1.08
Spambase	57	3677	1.54
Kr-vs-kp	37	2556	1.09
Dermatology	34	358	16.9
Segment0	19	2308	6.02
Vehicle0	18	846	3.25
Vehicle1	18	846	2.90
Vehicle2	18	846	2.88
Vehicle3	18	846	2.90
Page-blocks0	10	5472	8.79
Wisconsin	9	683	1.86
Pima	8	768	1.87
Yeast1	8	1484	2.46
Yeast3	8	1484	8.10
Ecoli-0_vs_1	7	220	1.86
New-thyroid1	5	215	5.14

[1] http://keel.es/
[2] http://archive.ics.uci.edu/ml/

5.1.3. Resultados en problemas de clasificación

Resultados experimentales con LIA:

En la Figura 5.1 se muestra el efecto que tiene el parámetro *depth* (para los valores 1, 2 y 3) en los resultados obtenidos por LIA. *Opt* indica la la optimización descrita en la Sección 3.3.2, consiste en procesar solo los subconjuntos de 3 atributos donde al menos dos permitan encontrar información cuando son procesados en parejas. La figura está dividida en tres secciones verticales correspondientes a los resultados para cada clasficador: J48, K-NN y MLP. En cada sección, en el eje X se representa el porcentaje de atributos seleccionados y en el eje Y el promedio normalizado de *G-mean*. Para todos los clasificadores, los mejores resultados se obtienen en orden al valor de *depth*, mejor para $depth = 3$ y peor para $depth = 1$. Es destacable que los resultados para $depth = 3$ optimizado son muy similares (a excepción de en MLP para el 30 % de atributos) a los obtenidos para $depth = 3$ sin optimizar, siendo esta última opción mucho más lenta.

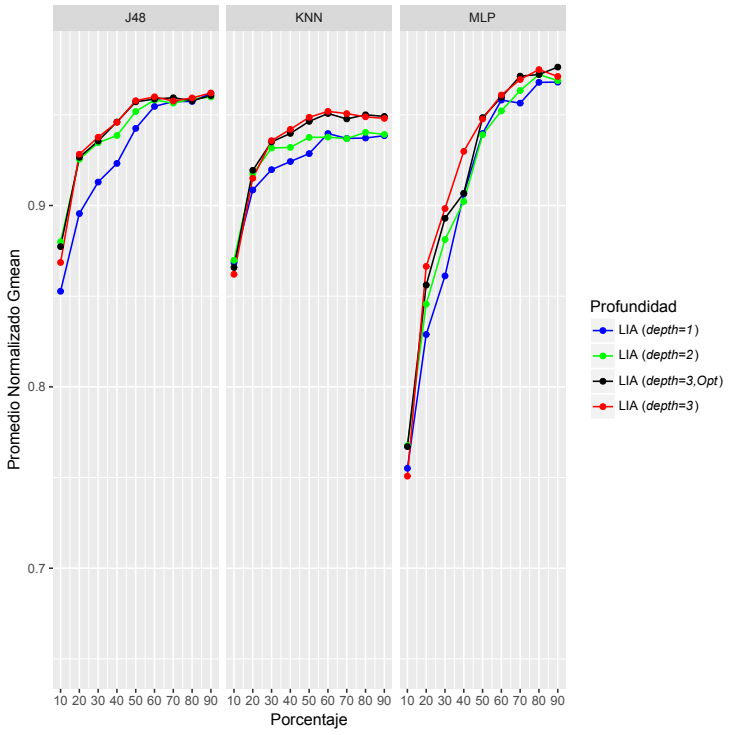

Figura 5.1: Dominios de clasificación - LIA con diferentes valores de *depth*

En la Figura 5.2 se compara LIA($depth = 3$), la opción que mejores resultados ha dado en LIA, con el resto de técnicas de selección. Claramente puede observarse como

LIA($depth = 3$) domina al resto de técnicas para cualesquier tanto por ciento de atributos seleccionados y con las tres técnicas de clasificación utilizadas. Puede observarse, también, como CFS-Subset obtiene los peores resultados, seguido de ReliefF y Chi-Squared. Las diferencias a favor de LIA se hacen máximas, para todos los clasficadores, para el 20 % de los atributos.

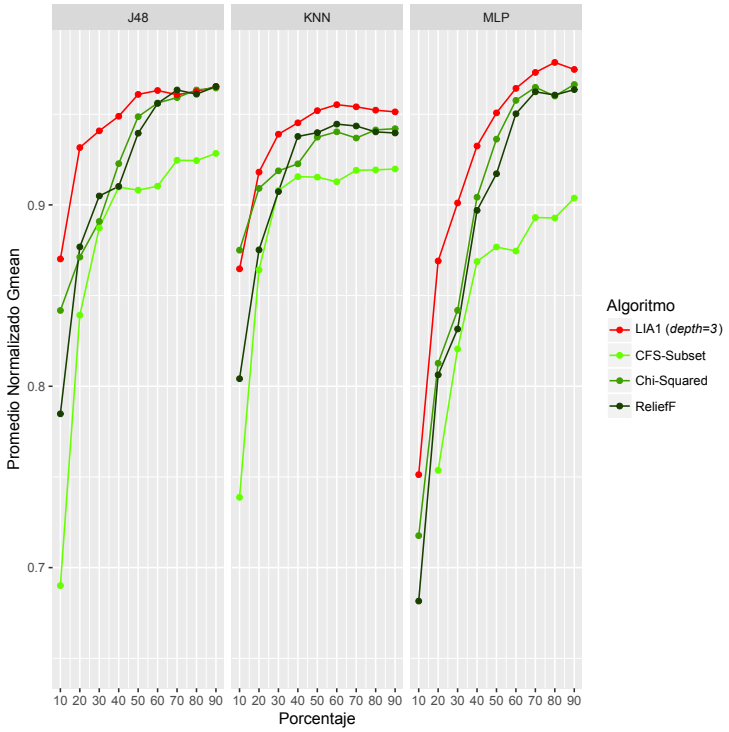

Figura 5.2: Dominios de clasificación - LIA($depth = 3$) *vs* Otros

En la Figura 5.3 se muestran los resultados del test de significancia (Wilcoxon) sobre los resultados obtenidos utilizando LIA frente a `CFS-Subset`, Chi-Squared y ReliefF. La figura contiene tres secciones verticales correspondientes a los resultados para cada clasificador: J48, K-NN y MLP. En cada sección, en el eje X se representan las técnicas de selección de atributos frente a las que comparar: CFS-Subset, ReliefF y Chi-Squared. En el eje Y, se muestran los porcentajes de atributos seleccionados. En las intesersecciones se dibujan 4 círculos, que corresponden a diferentes valores del parámetro *depth*. De izquierda a derecha cada círculo corresponde a: ($depth = 1$), ($depth = 2$), ($depth = 3, opt$) y ($depth = 3$). El color del círculo será blanco para indicar que la técnica a comparar ha dado un resultado mejor que LIA y negro para indicar que

LIA ha sido superior. Si el círculo, es grande indica que la mejora es estadísticamente significativa.

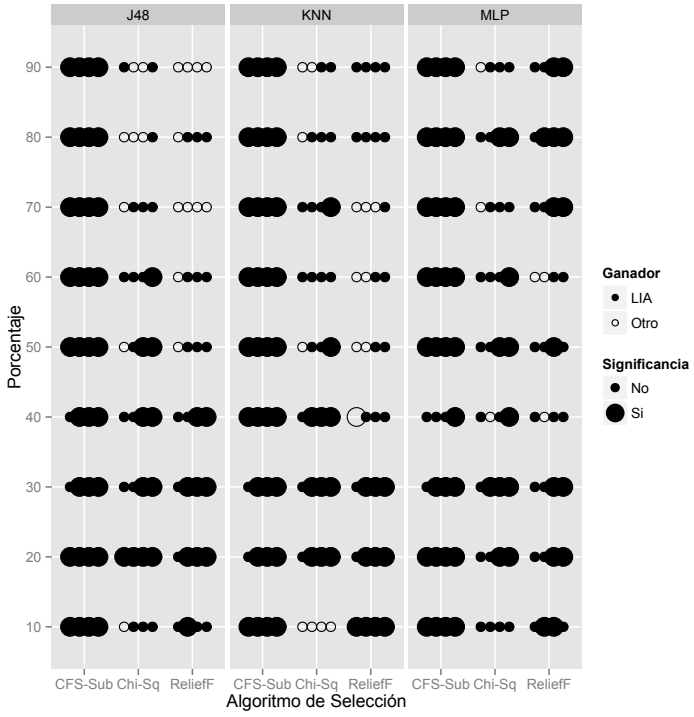

Figura 5.3: Dominios de clasificación - Significación estadística LIA *vs* Otros

Observando tanto la Figura 5.2 como la Figura 5.3, se pueden establecer las siguientes comparaciones entre cada técnica de selección de atributos y LIA:

- LIA *versus* `CFS-Subset`: LIA proporciona mejores resultados que CFS-Subset para todas las técnicas de clasificación y para todos los tantos por cientos de atributos. Además, la mejora puede considerarse estadísticamente significativa a excepción de unos pocos casos para $depth = 1$.
- LIA *versus* Chi-Squared: Con pocas excepciones, LIA supera casi siempre a Chi-Squared (89 ocasiones frente a 19). La ganancia a favor de LIA se manifiesta principalmente para menos del 50 % de atributos. A mayor profundidad de exploración (*depth*), mayor y en más ocasiones estadísticamente significativa es la ganancia de LIA frente a Chi-Squared. En ningún caso Chi-Squared supera a LIA de forma significativa.

- LIA *versus* ReliefF: La mejora de LIA se observa principalmente cuando se seleccionan pocos atributos, concretamente para menos del 50 %. Para el 50 % ó más de atributos ambas técnicas producen resultados bastante similares, a excepción de con MLP donde LIA supera también a ReliefF. Según aumenta *depth* los resultados de LIA mejoran frente a ReliefF, tanto en promedio de *G-mean* como en significancia estadística. ReliefF solo supera una vez a LIA de forma estadísticamente significativa (para el 40 % de atributos, haciendo uso de K-NN).

Resultados experimentales con LIA2:

Siguiendo el mismo procedimiento que el utilizado para mostrar los resultados de LIA se han generado las siguientes figuras: la Figura 5.4 que muestra el efecto del parámetro *depth* sobre LIA2 (en este caso *depth* ha tomado los valores 1, 2 y 3), la Figura 5.5 donde se compara LIA2($depth = 3$) con el resto de técnicas de selección y la Figura 5.6 donde se muestra el resultado de significación estadística de la comparación entre LIA2 y el resto de técnicas.

Puede observarse en la Figura 5.4 que para J48 y MLP los resultados para $depth = 2$ y $depth = 3$ son bastante similares, con ligera ventaja a favor de $depth = 3$. Los peores resultados se obtienen para $depth = 1$. Para K-NN, el uso de $depth = 3$ produce resultados superiores al resto para más del 30 % de atributos.

Puede observarse en la Figura 5.5 como LIA2($depth = 3$) en J48 y MLP, domina para cualesquier tanto por ciento de atributos. Para K-NN, Chi-Squared supera ligeramente a LIA2($depth = 3$) para menos del 40 % de atributos, siendo LIA2($depth = 3$) superior para más del 40 % de atributos.

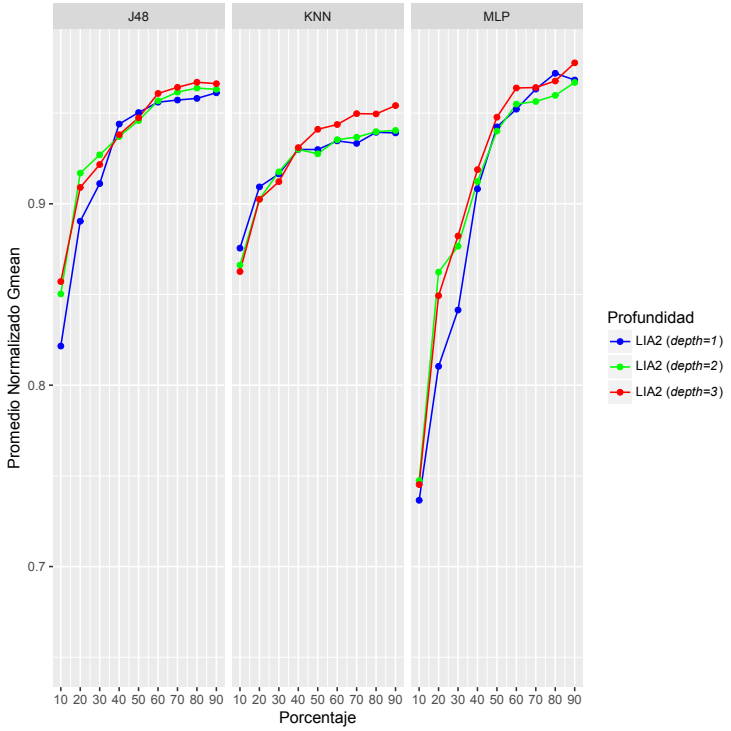

Figura 5.4: Dominios de clasificación - LIA2 con diferentes valores de *depth*

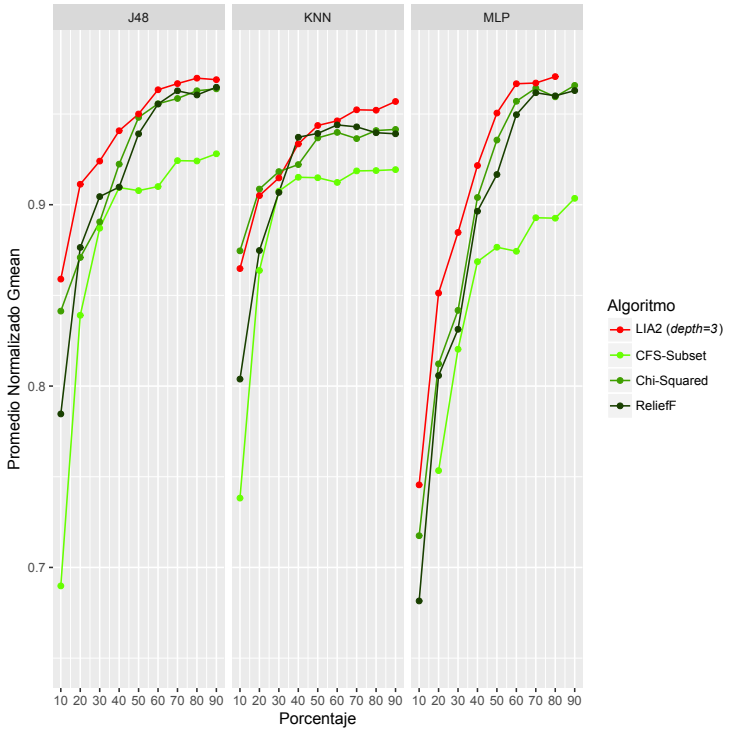

Figura 5.5: Dominios de clasificación - LIA2($depth = 3$) *vs* Otros

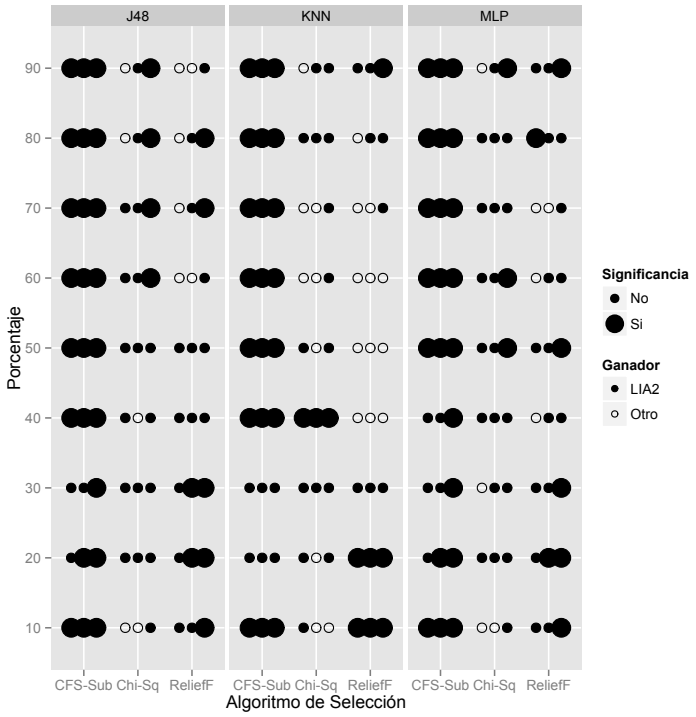

Figura 5.6: Dominios de clasificación - Significación estadística LIA2 *vs* Otros

Observando tanto la Figura 5.5 como la Figura 5.6, se pueden establecer las siguientes comparaciones entre cada técnica de selección de atributos y LIA2:

- LIA2 *versus* `CFS-Subset`: LIA2 proporciona resultados significativamente mejores para la gran mayoría de los clasificadores y porcentaje de atributos. Se observa una mejora, en cuanto significancia, para $depth = 3$.

- LIA2 *versus* Chi-Squared: LIA2 supera significativamente a Chi-Squared en varias ocasiones, mientras que Chi-Squared nunca es significativamente mejor que LIA2. Los mejores resultados para LIA2 se alcanzan para $depth = 3$. Para LIA2($depth = 3$) y observando los promedios normalizados de *G-mean*, LIA2 supera claramente a Chi-Squared para todos los porcentajes de atributos y técnica de clasificación utilizada.

- LIA2 *versus* ReliefF: En J48 y MLP, LIA2 obtiene mejores resultados que ReliefF especialmente para $depth = 3$. Para K-NN, LIA2 supera 6 veces a ReliefF y éste

supera en tres ocasiones a LIA2. No obstante, la mejora es significativa en tres ocasiones a favor de LIA2 y en ninguna para ReliefF.

Resultados experimentales con LIA3:

Siguiendo el mismo procedimiento que el utilizado para mostrar los resultados de LIA y LIA2 se han generado las siguientes figuras: la Figura 5.7 que muestra el efecto del parámetro *depth* sobre LIA3, la Figura 5.8 donde se compara LIA3($depth = 3$) con el resto de técnicas de selección y la Figura 5.9 donde se muestra el resultado de significación estadística de la comparación entre LIA3 y el resto de técnicas.

Puede observarse en la Figura 5.7 que el mejor resultado se obtiene para todas los clasificadores haciendo uso de $depth = 3$. Resultados intermedios se obtienen con $depth = 2$ y, con $depth = 1$ los peores. Solo se observa una excepción, para J48 y el 40 % de atributos, donde $depth = 3$ obtiene peores resultados que el resto, si bien, la diferencia es pequeña.

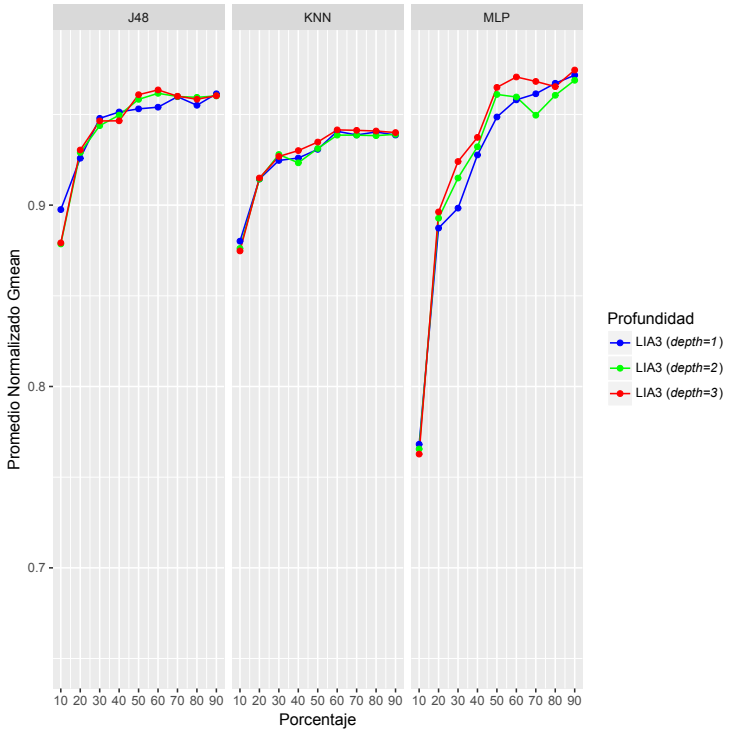

Figura 5.7: Dominios de clasificación - LIA3 con diferentes valores de *depth*

En la Figura 5.8 puede observarse como LIA3($depth = 3$) domina en J48 y MLP, para cualquier tanto por ciento de atributos. Con K-NN, Chi-Squared supera ligeramente a LIA3($depth = 3$) para más de 30 % y menos del 70 % de atributos, siendo LIA3($depth = 3$) superior en el resto.

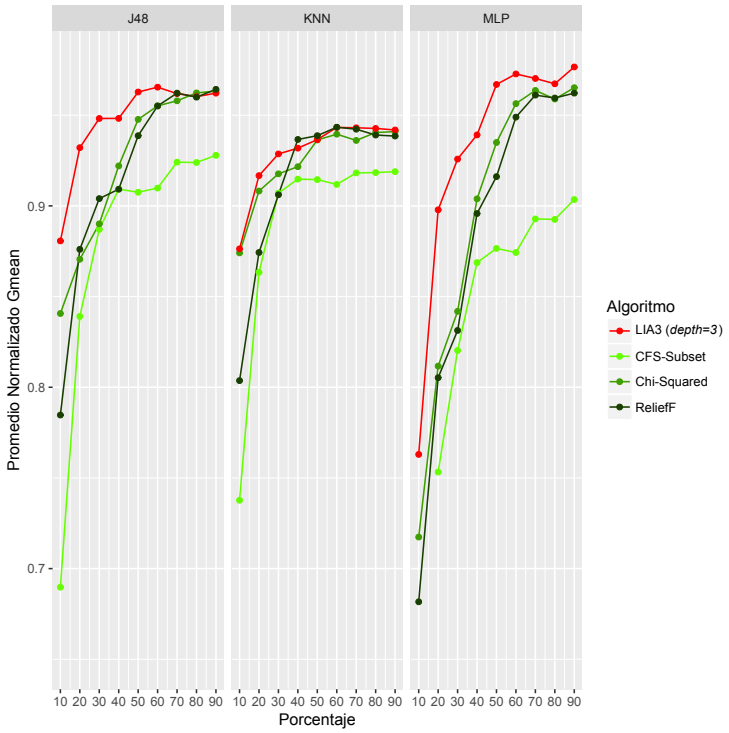

Figura 5.8: Dominios de clasificación - LIA3($depth = 3$) *vs* Otros

Observando tanto la Figura 5.8 como la Figura 5.9, se pueden establecer las siguientes comparaciones entre cada técnica de selección de atributos y LIA3:

- LIA3 *versus* `CFS-Subset`: LIA3 proporciona resultados significativamente mejores para todos los clasificadores, tanto por ciento de atributos y valor de *depth*.
- LIA3 *versus* Chi-Squared: En J48 y MLP, LIA3 supera significativamente a Chi-Squared especialmente para cuando el porcentaje de atributos seleccionados es menor de 50 %. Para el resto, LIA3 ofrece mejores resultados, pero no de forma estadísticamente significativa en todos los casos, aunque con $depth = 3$ los resultados mejoran sobre $depth = 1$. En el caso de K-NN, LIA3 parece mejor que Chi-Squared, pero los resultados están más igualados.

- LIA3 *versus* ReliefF: En J48 y MLP, LIA3 obtiene mejores resultados que ReliefF especialmente para $depth = 3$. Para K-NN, LIA3 supera de forma significativa a ReliefFcuando el porcentaje de atributos es menor que 40 %. Para K-NN y el 40 % de atributos ReliefF es superior a LIA3.

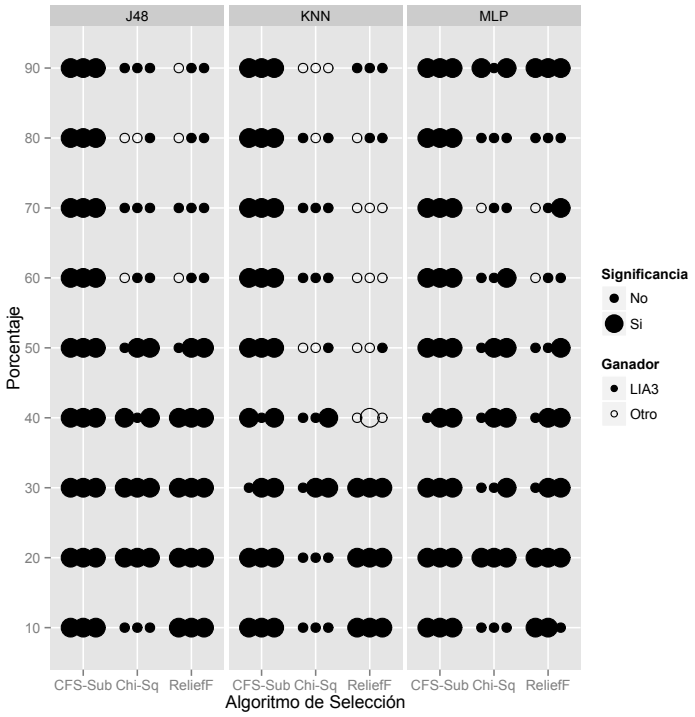

Figura 5.9: Dominios de clasificación - Significación estadística LIA3 *vs* Otros

Comparación entre las diferentes variantes de LIA:

Por último, en la Figura 5.10 se compara el desempeño de cada variante de LIA. Se comparan todas las variante de LIA parametrizadas para $depth = 3$, dado que es para este valor donde todas han ofrecido mejores resultados. En cada sección vertical se muestran los resultados para una técnica de clasificación (J48, K-NN o MLP). En el eje X de cada sección se representa el porcentaje de atributos seleccionados y en el eje Y el valor de la media normalizada de *G-mean*. Puede observarse como para J48 y MLP, LIA3 supera al resto excepto para más del 70 % de atributos (donde LIA2 obtiene el mejor resultado), LIA obtiene resultados bastante similares a LIA3, especialmente en J48. En el caso de K-NN es LIA la técnica que mejores resultados ofrece, hasta el 80 % de atributos (donde es superada por LIA2), LIA3 obtiene resultados similares a LIA hasta el 40 % de atributos.

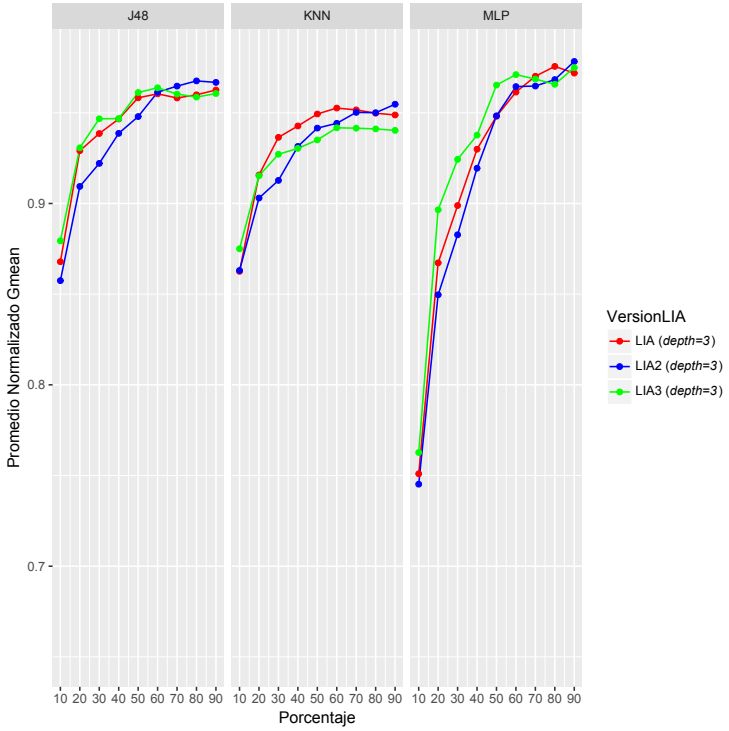

Figura 5.10: Dominios de clasificación - LIA *vs* LIA2 *vs* LIA3

5.1.4. Resumen de los resultados en problemas de clasificación

Se ha observado que todas las variantes de LIA obtienen resultados competitivos, tanto en promedio de *G-mean* como en significancia estadística, frente a las otras técnicas de selección: `CFS-Subset`, ReliefF y Chi-Squared, independientemente del clasificador. Las comparaciones utilizando el promedio de *G-mean* muestran (Figuras 5.2, 5.5 y 5.8) la superioridad de las variantes de LIA frente a CFS-Subset, ReliefF y Chi-Squared para todos los clasificadores utilizados. Desde el punto de la significación estadística, las tres variantes de LIA han mostrado (Figuras 5.3, 5.6 y 5.9) ser claramente superiores a `CFS-Subset`, mejores que Chi-Squared y superiores, aunque de forma más ajustada, frente a ReliefF.

Para las tres variantes de LIA se ha observado (Figuras 5.1, 5.4 y 5.7), también, la mejora de los resultados según aumenta la profundidad de exploración (parámetro *depth*). Así, los mejores se han obtenido en todos los casos para $depth = 3$.

En cuanto a la comparación entre las propias variantes de LIA, los experimentos han mostrado (ver Fig. 5.10) como, para J48 y MLP, hasta el 70 % de atributos LIA3 es la que mejores resultados obtiene, seguida de LIA. En el caso de K-NN la mejor técnica ha sido LIA hasta el 70 % de atributos, igualada por LIA3 hasta el 30 %. En cualquier caso, hasta el 20 % de atributos las tres variantes han obtenido resultados muy similares para todos los clasificadores y, para el 90 % de atributos LIA2 ha superado al resto de técnicas.

Eventualmente, también se ha observado que un mismo subconjunto de atributos seleccionados, haciendo uso de una técnica dada, obtiene resultados significativamente superiores al resto usando un clasificador y la situación se invierte usando otro distinto (por ejemplo, en las Figuras 5.9 y 5.8 ver resultados para el 40 % de atributos con ReliefF y K-NN). Ésto puede ser debido a que pueda existir cierta relación entre los selectores de atributos y los clasificadores, que haga que para cada clasificador sea más o menos apropiada un técnica de selección de atributos.

5.2. Dominios de regresión

5.2.1. Marco experimental

En esta sección se muestran los resultados de aplicar las técnicas de selección de atributos desarrolladas (LIA, LIA2 y LIA3) a problemas de regresión.

Para abordar este tipo de problemas se ha desarrollado un técnica, LIAR (ver Sección 3.7), que particiona el problema de regresión en un cierto número de problemas binarios, para cada problema binario hace uso de LIA, LIA2 o LIA3, como técnicas subordinadas, para obtener la lista de atributos relevantes y, finalmente, combina dichas listas para obtener los atributos relevantes para el proceso de regresión.

LIAR y todas las variantes de LIA han usado la parametrización por defecto, a excepción de LIA que se ha configurado, para estos experimentos, haciendo uso del parámetro $mode = 2$. Este modo de funcionamiento se ha diseñado con el objetivo de ordenar los atributos según la capacidad de cada atributo de aportar información (reducción de entropía) a la solución del problema, ya sea solo o en combinación con otros.

Se han realizado dos bloques de experimentos. Para el primero se han utilizado dominios de regresión obtenidos del repositorio público KEEL (*http://keel.es/*) y, para el segundo, se ha trabajado sobre un problema de predicción fotovoltaica publicado en Kaggle (*https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest*).

Para el **primer bloque experimental**, el rendimiento de cada variante de LIA se ha comparado frente a dos técnicas de selección de atributos: CFS-Subset y ReliefF, incluidas en la herramienta WEKA (Witten et al., 2011). Para una descripción de cada técnica consultar Capítulo de Estado de la Cuestión (ver Sección 2.4.1). En este caso, no se ha utilizado la técnica Chi-Squared dado que los dominios de regresión presentan atributos numéricos y Chi-Squared requiere que los datos de entrada estén discretizados.

Para CFS-Subset y ReliefF todos los parámetros han tomado el valor por defecto establecido en WEKA. Como algoritmos de búsqueda asociados a CfsSubsetEval se han utilizado: ExhaustiveSearch para dominios de menos de 25 atributos y, BestFirst para el resto.

Para evaluar la lista de atributos obtenidos por cada técnica de selección de atributos, se han utilizado éstos con la técnica de regresión SMOreg (ver Sección 2.4.3),

también incluida en WEKA. SMOreg ha sido parametrizada según valores por defecto, a excepción de *kernnel='RBFKernel'* y *RegOptimizer='SMOImproved'*.

La medida utilizada para comparar la bondad de la regresión ha sido *Square Mean Error* (SME), Ecuación 5.2. Siendo y el vector de salida real y p el vector de predicciones.

$$SME(y,p) = \frac{\sum_{k=1}^{n}(y_k - p_k)^2}{n} \tag{5.2}$$

El procedimiento seguido ha sido similar al utilizado en los experimentos de clasificación (ver Sección 5.1.1), utilizando en este caso SME y no *G-mean*.

En las figuras donde se presentan los resultados se usa el Promedio Normalizado de *SME*. Para calcular éste se procede de la siguiente manera: para cada dominio, una vez hecha la media de los 5 $folds$, se calcula el menor SME obtenido (para cualquier tanto por ciento de atributos y algoritmo) y se utiliza para normalizar el resto de resultados. Así, en cada dominio, todos los resultados estarán en el rango [1.. ínf]. A continuación, se calcula el promedio para todos los dominios, utilizando para cada uno los valores normalizados. Valores menores de promedio de *SME* indican mejor desempeño.

Para el **segundo bloque experimental**, el rendimiento de cada variante de LIA se ha comparado frente a dos técnicas de selección de atributos: ReliefF y Correlación Lineal incluidas en WEKA (ver Sección 2.4.1). En este caso, no se ha utilizado la técnica Chi-Squared dado que los dominios de regresión presentan atributos numéricos y Chi-Squared requiere que los datos de entrada estén discretizados. CFS-Subset tampoco se ha utilizado este segundo bloque dado que el problema de predicción fotovoltaica cuenta con 1200 atributos y CFS-Subset ha resultado una técnica prohibitivamente lenta para afrontar estos problemas.

Todos los parámetros de ReliefF se han configurado con el valor por defecto establecido en WEKA.

Para evaluar la lista de atributos obtenidos por cada técnica de selección de atributos se ha utilizado, en este segundo bloque, la técnica de regresión Gradient Boosted Regression (GBR) (ver Sección 2.4.3), incluida en R (Team, 2014). La técnica GBR se ha configurado con los siguientes parámetros: $Trees = 5000$, $shrinkage = 0.01$ y $treeDepth = 10$. Dichos valores han sido calculados siguiendo el procedimiento descrito en (**?**).

La medida utilizada para comparar la bondad de la regresión ha sido, en este segundo bloque, *Mean Absolute Error* (MAE), Ecuación 5.3. Siendo y el vector de salida real y p el vector de predicciones.

$$MAE(y,p) = \frac{\sum_{k=1}^{n} |y_k - p_k|}{n} \tag{5.3}$$

En este segundo bloque no se ha utilizado validación cruzada. Dado que se dispone de 98 parejas de ficheros de entrenamiento y test, correspondientes a las 98 estaciones, los valores mostrados en las figuras corresponden al promedio de MAE.

5.2.2. Descripción de los dominios de regresión

Para este experimento de selección de atributos, para problemas de regresión, se ha preparado un conjunto de problemas, obtenido del repositorio público KEEL (KEEL-dataset repository)[1] (Alcalá-Fdez et al., 2011). La la Tabla 5.2 muestra el nombre de cada problema, el número de instancias de cada problema y el número de atributos del mismo. KEEL pone a disposición del experimentador varios conjuntos de datos. El elegido para este experimento ha sido el particionado para procesos de validación cruzada, con 5-folds extratificados, por problema.

Tabla 5.2: Dominios para regresión (KEEL)

Nº	Dominio	Nº Instancias	Nº Atributos
1	abalone	4.177	8
2	california	20.540	8
3	compactiv	8.192	21
4	concrete	1.030	8
5	house	22.784	16
6	mortgage	1.049	15
7	mv	40.755	10
8	pole	14.998	26
9	puma32h	8.192	32
10	stock	950	9
11	treasury	1.049	15
12	wankara	1.509	9

[1] http://keel.es/

Otro segundo bloque de problemas de regresión, utilizados para validar las nuevas técnicas de selección de atributos, en problemas de regresión, se ha obtenido a partir de un problema real publicado en Kaggle [1]. Básicamente, el desafío consiste en predecir la producción eléctrica de 98 plantas fotovoltaicas en el estado de Oklahoma, a partir de datos metereológicos, ofrecidos por la American Meteorological Society, sobre un territorio de 180.000 km^2. Para cada planta, se ha generado un problema con 1200 atributos (16 estaciones cercanas con 75 variables cada una, $16 \times 75 = 1200$) y un valor de clase de salida consistente en un número real que contiene el valor a predecir.

5.2.3. Resultados en problemas de regresión (KEEL)

Resultados experimentales con LIA:

En la Figura 5.11 se muestran los resultados obtenidos haciendo uso de LIAR (con LIA como técnica subordinada), para *depth=1, 2, 3 y 3 optimizada*. En el eje X se representa el porcentaje de atributos seleccionado y en el eje Y el promedio del error normalizado. Los mejores resultados se obtienen en orden a la profundidad de exploración (*depth*), mejor para $depth = 3$ y peor para $depth = 1$. A partir del 50 % de atributos los resultados son muy similares para cualquier valor de *depth*.

En la Figura 5.12 se compara LIA($depth = 3$) con CFS-Subset y ReliefF. Puede observarse como LIA domina para cualquier tanto por ciento de atributos a excepción de 20 % y 70 % donde iguala a ReliefF. El peor comportamiento lo presenta `CFS-Subset`.

En la Figura 5.13 se muestran los resultados del test de significancia (Wilcoxon) sobre los resultados obtenidos utilizando LIA frente a CFS-Subset y ReliefF. La figura contiene dos secciones verticales. La sección izquierda corresponde a CFS-Subset y la derecha a ReliefF. En cada intesersección se dibujan 4 círculos, que corresponden a diferentes valores del parámetro *depth*. De izquierda a derecha cada círculo corresponde a: ($depth = 1$), ($depth = 2$), ($depth = 3, opt$) y ($depth = 3$). El color del círculo será blanco para indicar que la técnica a comparar ha dado un resultado mejor que LIA y negro para indicar que LIA ha sido superior. Si el círculo, es grande indica que la mejora es estadísticamente significativa. Podemos observar como CFS-Subset es superior a LIA en 3 ocasiones (nunca significativamente), para el 20 % de los atributos, frente a 33 veces donde LIA es superior, en 22 de las cuales significativamente. LIA($depth = 3$) obtiene los mejores resultados superando a CFS-Subset en el 100 % de los casos, aunque solo

[1] https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest

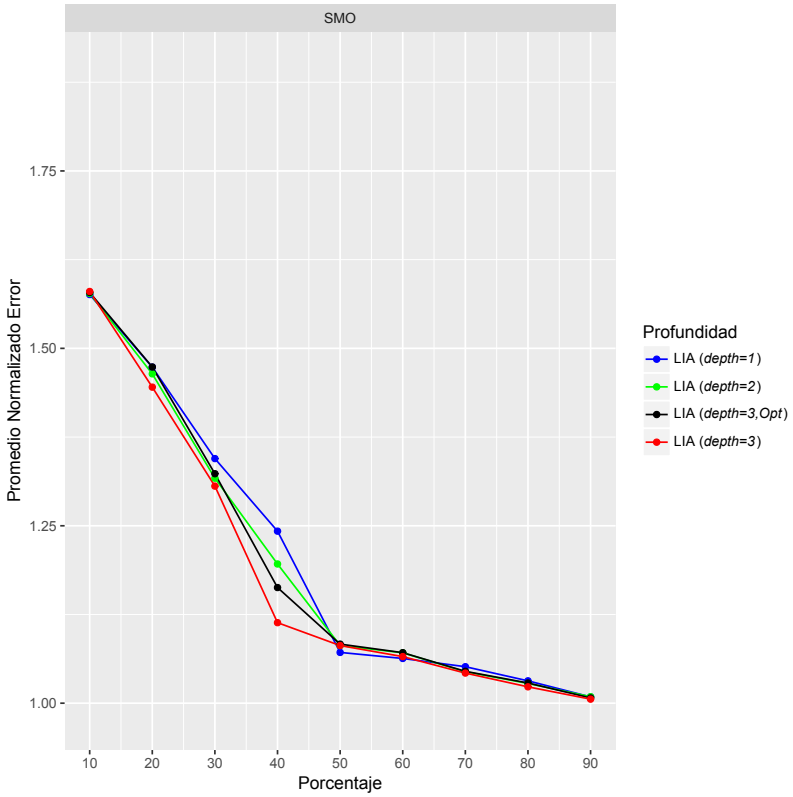

Figura 5.11: Dominios de regresión - LIA con diferentes valores de *depth*

6 veces de forma significativa. En cuanto a la comparación con ReliefF, se observa que LIA supera en 31 ocasiones a ReliefF, 6 de las cuales significativamente y, ReliefF supera a LIA en solo 5 ocasiones nunca significativas. Los mejores resultados de LIA se obtienen para el 10 % de los atributos y para ReliefF para el 70 %. Comparando frente a CFS-Subset y ReliefF los peores resultados se obtienen para $depth = 1$, si bien, las diferencias no resultan importantes.

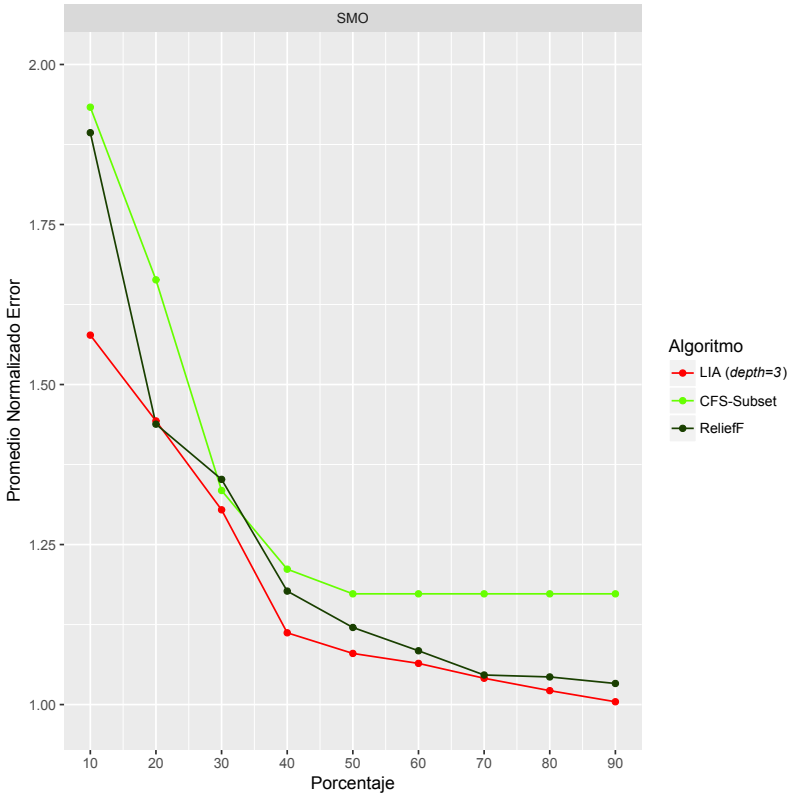

Figura 5.12: Dominios de regresión - LIA($depth = 3$) *vs* CFS-Subset y ReliefF

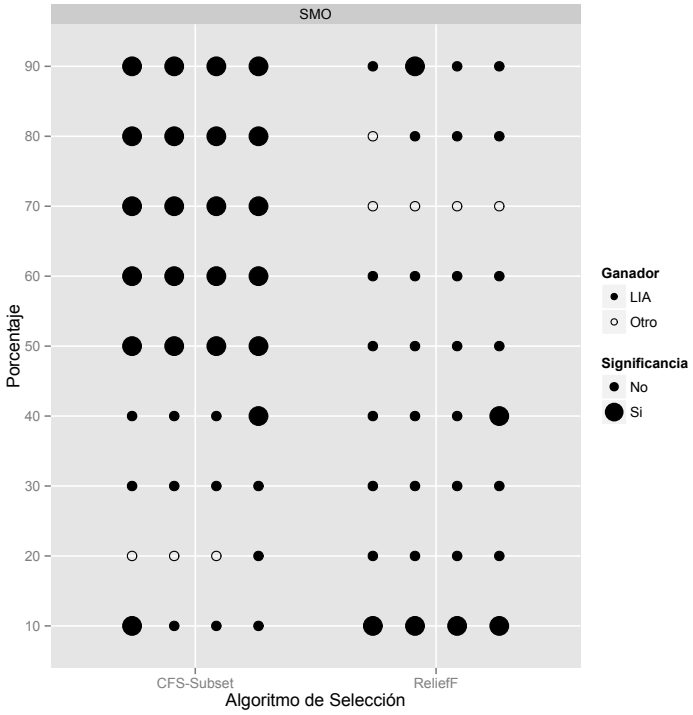

Figura 5.13: Dominios de regresión - Significación estadística LIA *vs* Otros

Resultados experimentales con LIA2:

Siguiendo el mismo procedimiento que para LIA, en la Figura 5.14 se analiza el efecto del parámetro *depth* en la técnica LIAR (con LIA2 como técnica subordinada). En este caso, *depth* toma los valores 1, 2 y 3.

Puede observarse como $depth = 1$ obtiene los peores resultados. Para $depth = 2$ y $depth = 3$ los resultados son mejores. En cualquier caso, todos muy similares.

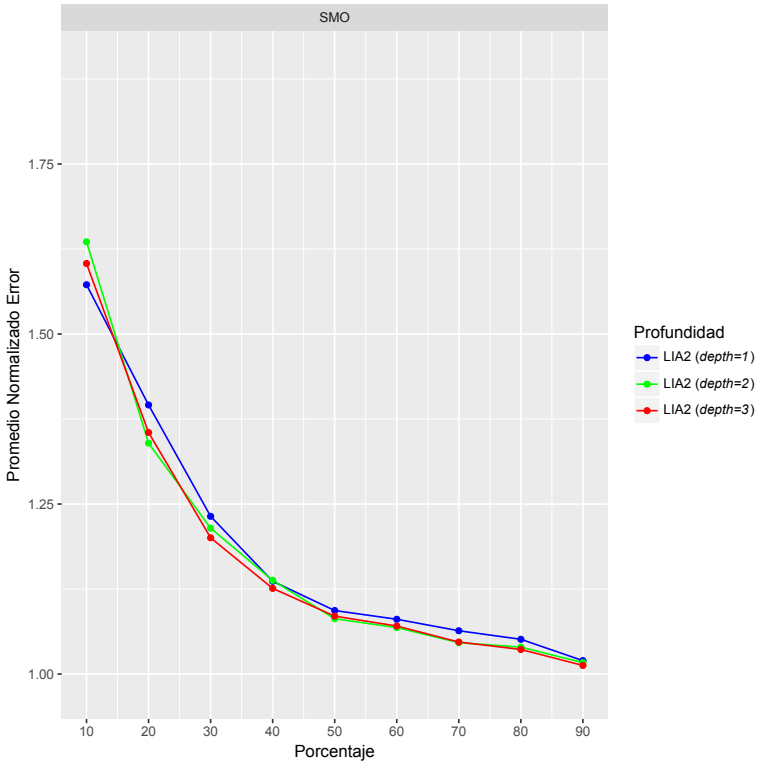

Figura 5.14: Dominios de regresión - LIA2 con diferentes valores de *depth*

En la Figura 5.15 se compara LIA2($depth = 3$) con CFS-Subset y ReliefF. Puede observarse como LIA2 domina para cualesquier tanto por ciento de atributos a excepción de para el 60 %, 70 % y 80 % donde los resultos son muy similares a ReliefF.

En la Figura 5.16 se muestran los resultados del test de significancia de los resultados de LIA2 frente a CFS-Subset y ReliefF. La figura se estructura de forma similar a la explicada para analizar la significancia de los resultados de LIA (ver Fig. 5.13). En este caso, en cada intesersección se dibujan 3 círculos, que corresponden a de izquierda a derecha con: ($depth = 1$), ($depth = 2$) y ($depth = 3$).

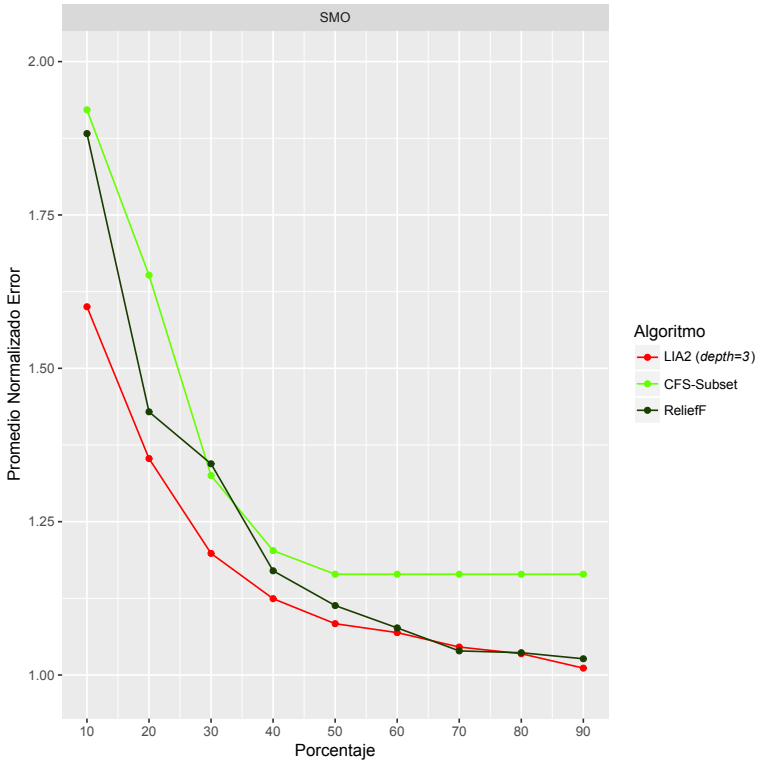

Figura 5.15: Dominios de regresión - LIA2($depth = 3$) *vs* CFS-Subset y ReliefF

Podemos observar como CFS-Subset solo es superior a LIA2 en 4 ocasiones (nunca significativamente) frente a 32 veces donde LIA2 es superior, en 17 de las cuales de forma estadísticamente significativa. Los mejores resultados de `CFS-Subset`, frente a LIA2, se obtienen principalmente para el 20 % de los atributos. En cuanto a la comparación con ReliefF, se observa que LIA2 supera en 31 ocasiones a ReliefF, 3 de las cuales significativas y, ReliefF supera a LIA2 en solo 5 ocasiones y nunca significativamente. Los mejores resultados para LIA2 se obtienen para $depth = 3$ seguidos, con poca diferencia, por $depth = 2$ y $depth = 3$.

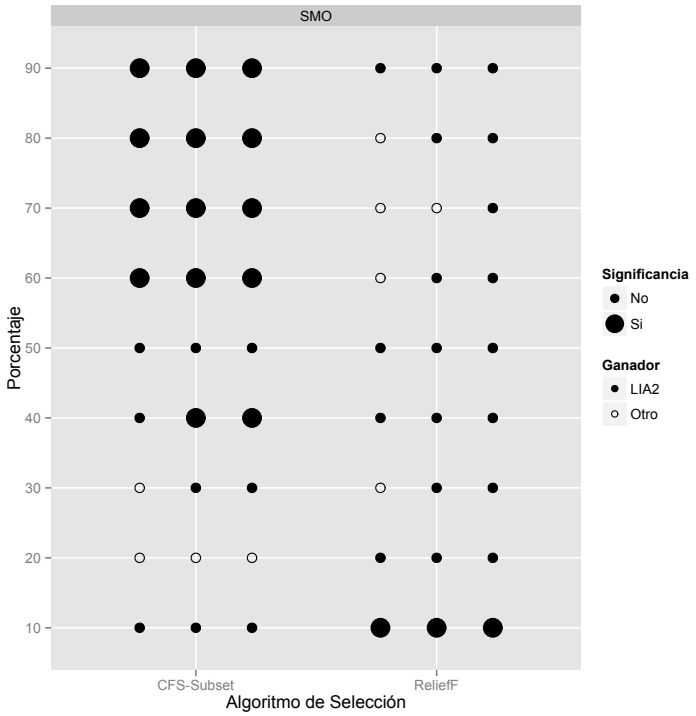

Figura 5.16: Dominios de regresión - Significación estadística LIA2 *vs* Otros

Resultados experimentales con LIA3:

Siguiendo el mismo procedimiento que para LIA y LIA2 en la Figura 5.17 se analiza el efecto del parámetro *depth* en la técnica LIAR (con LIA3 como técnica subordinada). Los resultados son muy similares para cualquier valor de *depth*.

En la Figura 5.18 se compara LIA3($depth = 3$) con CFS-Subset y ReliefF. Puede observarse como LIA3 obtiene mejores resultados que el resto de técnicas para cualquier tanto por ciento de atributos.

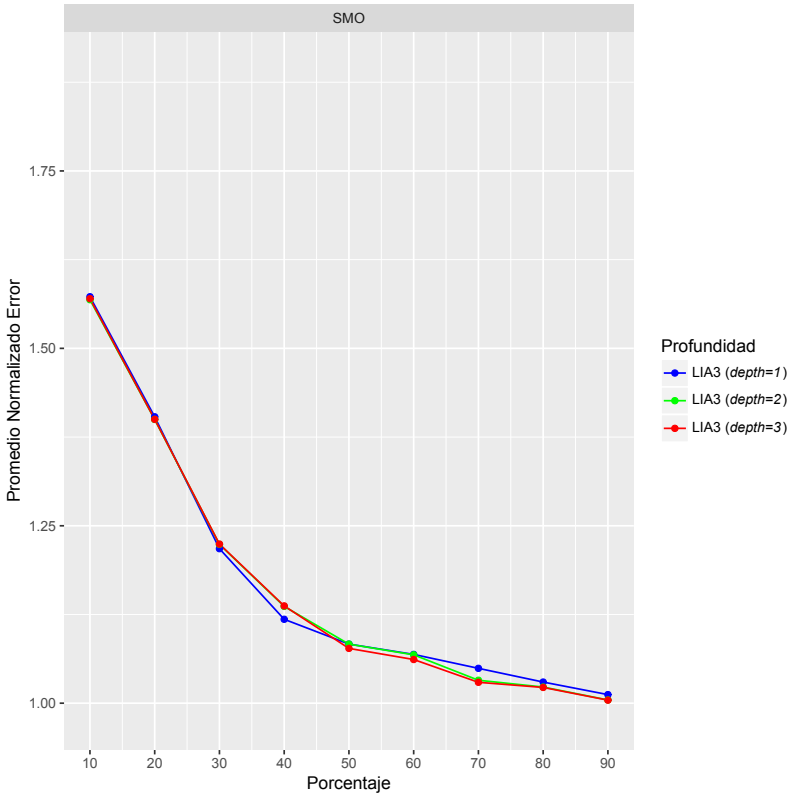

Figura 5.17: Dominios de regresión - LIA3 con diferentes valores de *depth*

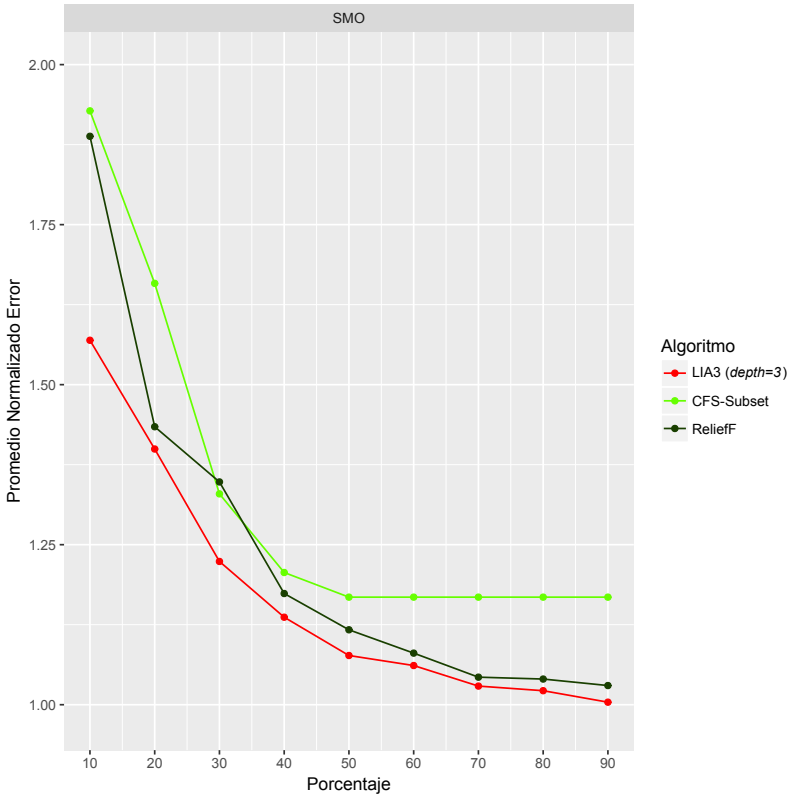

Figura 5.18: Dominios de regresión - Significación estadística LIA3 *vs* Otros

En la Figura 5.19 se muestran los resultados del test de significancia de los resultados de LIA3 frente a CFS-Subset y ReliefF. La figura se estructura de forma similar a la explicada para analizar la significancia de los resultados de LIA2 (ver Fig. 5.16). Podemos observar como CFS-Subset solo es superior a LIA3 en 6 ocasiones (nunca significativamente) frente a 30 veces donde LIA3 es superior, en 16 de las cuales de forma significativa. Los mejores resultados de `CFS-Subset`, frente a LIA3, se obtienen para el 20 % y el 30 % de los atributos. En cuanto a la comparación con ReliefF, se observa que LIA3 supera siempre a ReliefF, en 8 veces de forma estadísticamente significativa. Los mejores resultados para LIA3 se obtienen para $depth = 2$ y $depth = 3$ pero por muy poca diferencia respecto a $depth = 1$.

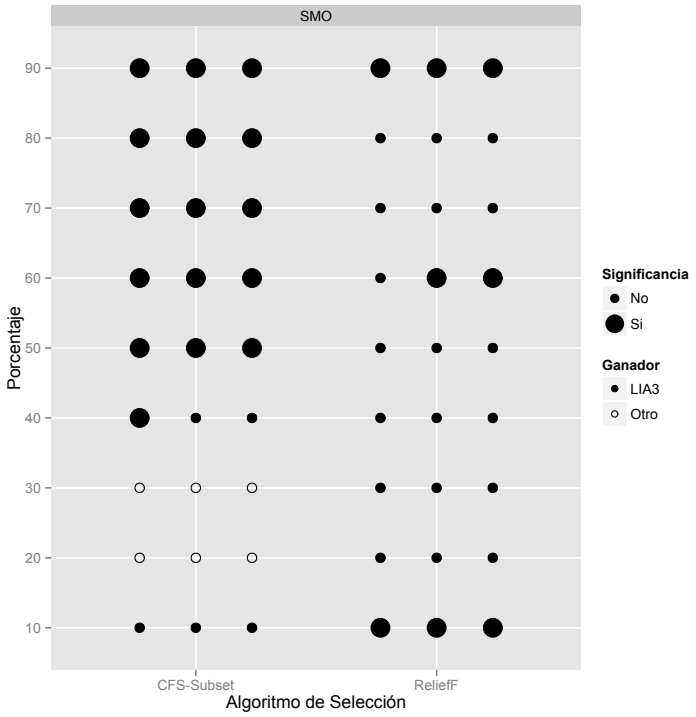

Figura 5.19: Dominios de regresión - Significancia LIA3 *vs* CFS-Subset y ReliefF

Comparativa en cada variante de LIA:

Por último, en la Figura 5.20 se comparan los resultados obtenidos con cada variante de LIA para $depth = 3$. Puede observarse como para menos del 50 % de atributos LIA es la que peor se comporta y LIA2 y LIA3, ofrecen en dicho rango un resultado muy

similar con ligera ventaja para LIA2. Para el 50 % o más de atributos todas las variantes ofrecen resultados muy similares, siendo LIA2 ligeramente peor que el resto.

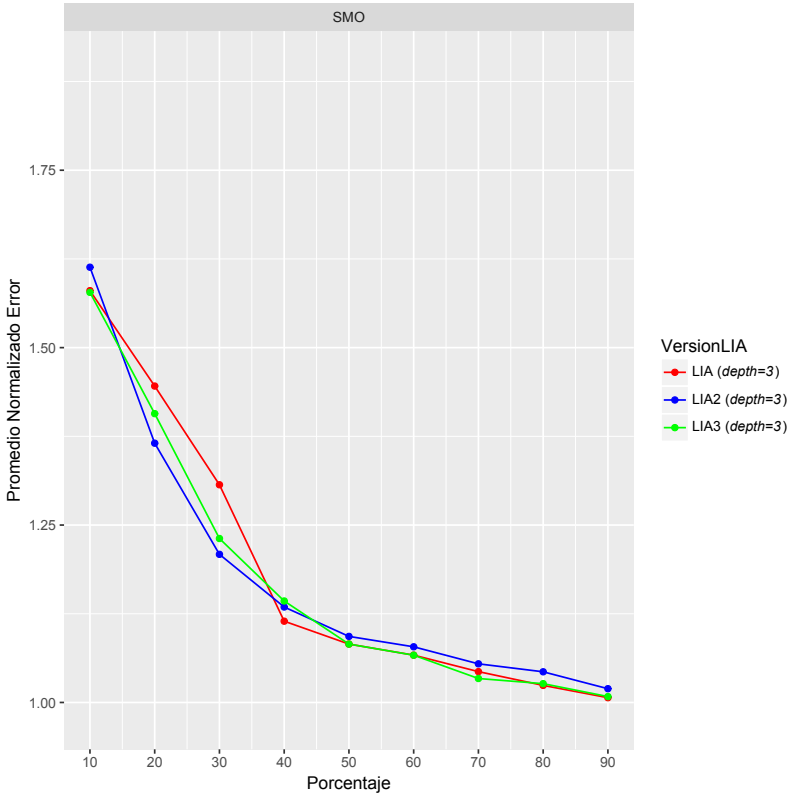

Figura 5.20: Dominios de regresión - LIA *vs* LIA2 *vs* LIA3

5.2.4. Resultados en problema de predicción fotovoltaica

A continuación, se detallan los resultados experimentales sobre el dominio SOLAR.

Resultados experimentales con LIA:

En la Tabla 5.3 se muestran los resultados obtenidos haciendo uso, como técnicas de selección de atributos, de Correlación Lineal, ReliefF y LIAR (con LIA como técnica subordinada), para *depth=1, 2 y 3*. Los valores mostrados son el promedio de MAE obtenido para cada número de atributos. Los resultados para Correlación Lineal son los peores (mayor valor de MAE) para todo el rango de atributos.

En la Figura 5.21 se muestran los resultados para LIA y para ReliefF. Los resultados para Correlación Lineal no se muestran aquí para mejorar la visualización del resto de

series. Como puede observarse ReliefF solo supera a LIA para 800 atributos y por una diferencia no demasiado importante. Con LIA los mejores resultados se obtienen con $depth = 2$ seguido de $depth = 3$.

Tabla 5.3: Dominio SOLAR - LIA

Atributos	C.Lineal	ReliefF	LIA($depth = 1$)	LIA($depth = 2$)	LIA($depth = 3$)
400	2.025.521	1.957.904	1.964.404	1.947.379	1.948.022
500	2.017.765	1.948.892	1.945.522	1.940.596	1.942.848
600	2.010.060	1.940.755	1.937.220	1.935.282	1.935.665
700	1.994.928	1.932.001	1.935.049	1.933.117	1.931.638
800	1.981.869	1.926.369	1.929.992	1.928.028	1.929.790
900	1.974.109	1.925.906	1.929.074	1.923.913	1.925.032
1000	1.942.152	1.924.534	1.925.281	1.924.931	1.923.867
1100	1.924.513	1.923.028	1.923.051	1.922.268	1.922.547

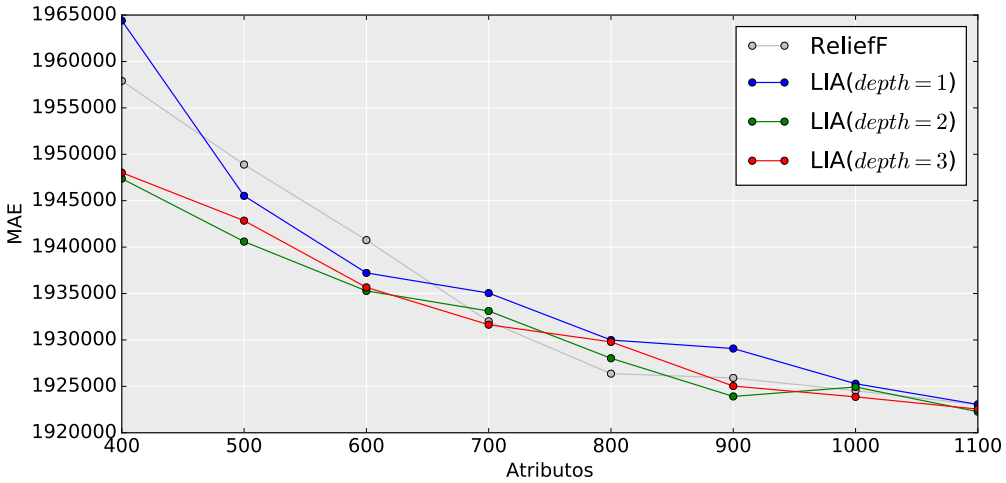

Figura 5.21: Dominio SOLAR - LIA

Resultados experimentales con LIA2:

Los resultados obtenidos para LIAR (con LIA2 como técnica subordinada) se muestran en la Tabla 5.4 y en la Figura 5.22.

Como puede observarse, los mejores resultados absolutos se obtienen para $depth = 3$ y, en menor medida, para $depth = 2$ solo superado por ReliefF para 800 atributos. En ambos casos, la diferencia con ReliefF es especialmente relevante en el rango de 400 a

700 atributos.

Tabla 5.4: Dominio SOLAR - LIA2

Atributos	C.Lineal	ReliefF	LIA2($depth = 1$)	LIA2($depth = 2$)	LIA2($depth = 3$)
400	2.025.521	1.957.904	1.978.836	1.933.377	1.932.183
500	2.017.765	1.948.892	1.960.392	1.928.918	1.930.166
600	2.010.060	1.940.755	1.950.324	1.931.003	1.926.085
700	1.994.928	1.932.001	1.943.495	1.926.836	1.924.193
800	1.981.869	1.926.369	1.938.655	1.927.651	1.923.142
900	1.974.109	1.925.906	1.934.920	1.927.030	1.925.782
1000	1.942.152	1.924.534	1.928.200	1.925.353	1.925.101
1100	1.924.513	1.923.028	1.923.818	1.922.549	1.922.144

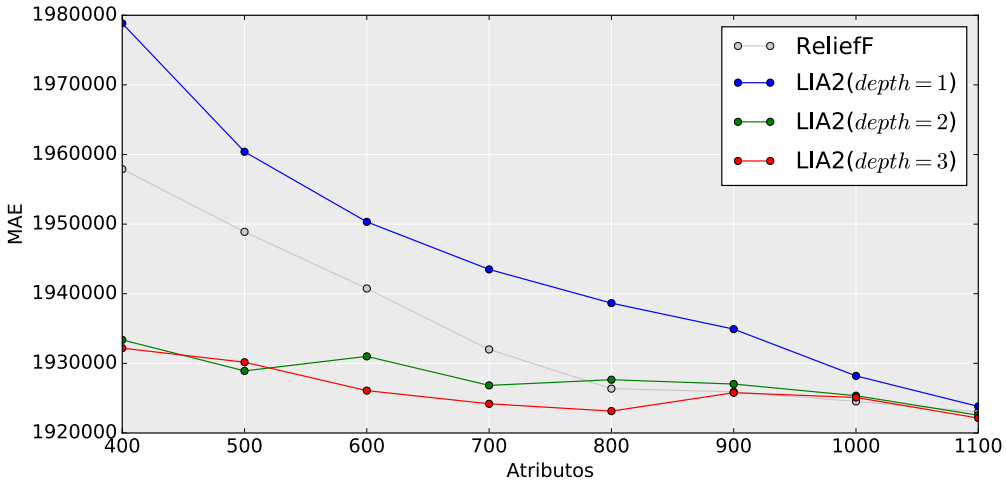

Figura 5.22: Dominio SOLAR - LIA2

Resultados experimentales con LIA3:

Los resultados obtenidos para LIAR (con LIA3 como técnica subordinada) se muestran en la Tabla 5.5 y en la Figura 5.23.

Puede observarse como los resultados son muy similares para todos los valores del parámetro *depth*. En general, los resultados son mejores que los obtenidos con ReliefF con la excepción del obtenido para 800 atributos donde ReliefF es mejor.

Tabla 5.5: Dominio SOLAR - LIA3

Atributos	C.Lineal	ReliefF	LIA3($depth = 1$)	LIA3($depth = 2$)	LIA3($depth = 3$)
400	2.025.521	1.957.904	1.949.518	1.949.501	1.951.436
500	2.017.765	1.948.892	1.941.310	1.942.412	1.942.143
600	2.010.060	1.940.755	1.933.548	1.936.064	1.934.000
700	1.994.928	1.932.001	1.931.104	1.932.378	1.931.931
800	1.981.869	1.926.369	1.927.813	1.930.482	1.927.589
900	1.974.109	1.925.906	1.925.325	1.925.666	1.926.460
1000	1.942.152	1.924.534	1.923.778	1.923.052	1.925.306
1100	1.924.513	1.923.028	1.923.773	1.921.789	1.923.498

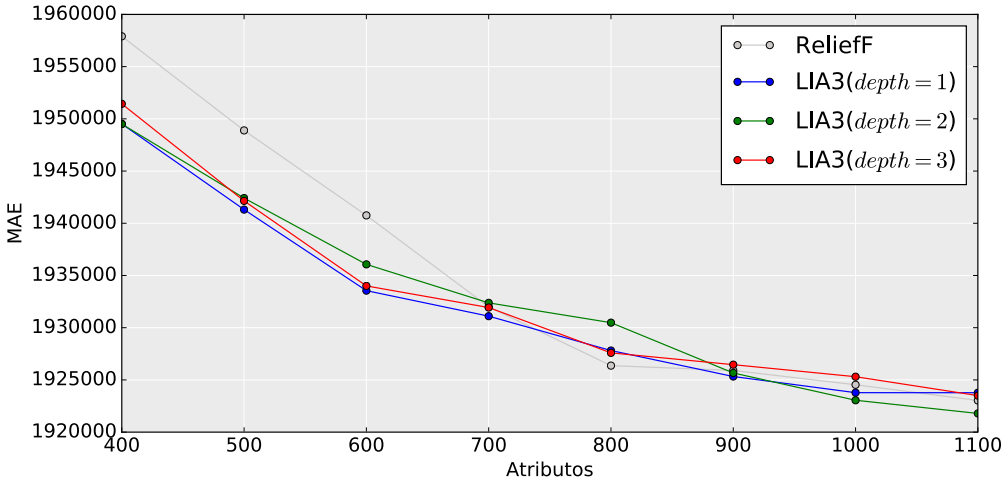

Figura 5.23: Dominio SOLAR - LIA3

Por último, en la Figura 5.24 se comparan las tres variantes de LIA, para el mejor valor de *depth*, junto con ReliefF. Como puede observarse, todas las variantes de LIA son competitivas frente ReliefF en este problema. LIA2 obtiene los mejores resultados, seguido de LIA3 y por último LIA. Cabe destacar el rendimiento de LIA2 para $atributos < 800$.

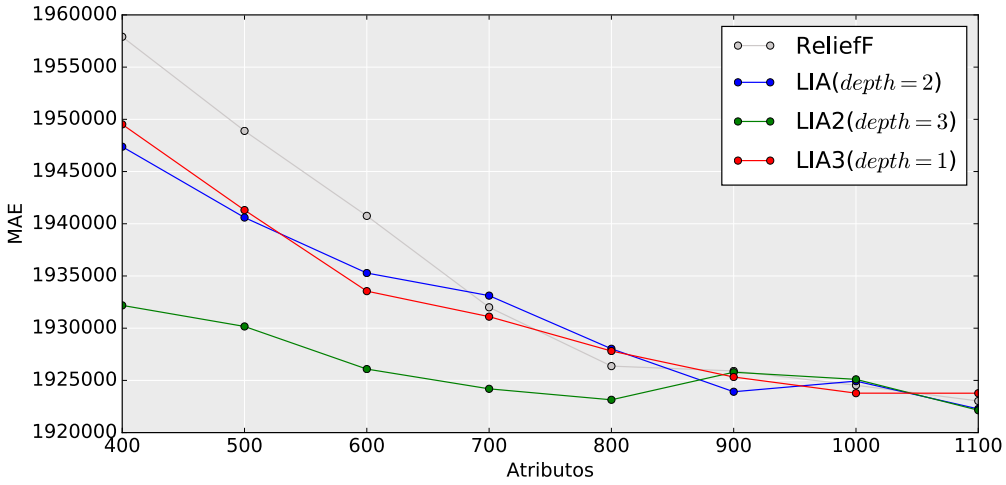

Figura 5.24: SOLAR - Comparación entre las diferentes implementaciones

5.2.5. Resumen de los resultados para problemas de regresión

En este bloque experimental se ha estudiado el comportamiento de las variantes de LIA y de LIAR, en problemas de regresión (obtenidos de KEEL y, en el caso de la aplicación fotovoltaica de Kaggle). Ha podido constatarse como, en dichos problemas, LIAR es competitiva frente a `CFS-Subset`, ReliefF y Correlación Lineal, obteniendo frecuentemente SME y MAE menores al resto de técnicas. Además, puede apreciarse que las técnicas LIA y LIA2 mejoran sus resultados según aumenta la profundidad de exploración (*depth*). En el caso de LIA3 no se aprecia dicha mejora. Dicho comportamiento puede explicarse por la semántica de *depth*: en el caso de LIA y LIA2, indica profundidad de exploración y dimensión de las soluciones encontradas (ej. para $depth = 2$, encuentran soluciones de hasta 2 atributos), en el caso de LIA3, se mantiene la semántica como profundidad de exploración (profundidad hasta la que LIA3 explora soluciones para decidir un criterio de partición en un nodo) pero no así la dimensión de cada solución final (profundidad total del árbol generado) que puede ser arbitrariamente alta.

Los experimentos han mostrado como, hasta el 40 % de los atributos seleccionados, LIA2 ha sido la técnica más eficaz, seguida de LIA3. A partir del 40 % la mejor técnica ha sido LIA, seguida de LIA3. Un comportamiento muy similar se ha observado en los experimentos Solar: hasta 800 atributos LIA2 ha sido la mejor, seguida de LIA3 y para

más de 800 atributos LIA ha obtenido buenos resultados, seguida de LIA3.

Por todo ello, en problemas de regresión, parece recomendable el uso de LIA2 para seleccionar pocos atributos, menos del 40 % y LIA para seleccionar más del 40 %. LIA3, sin ser la mejor en ningún rango de atributos seleccionados, ha sido la técnica más constante, quedando en segunda posición para cualquier porcentaje de atributos seleccionado.

6

Conclusiones y futuros trabajos

Predecir los movimientos futuros del mercado de valores a partir de los datos históricos, es un problema que todavía presenta cierta controversia. Una revisión bibliográfica preliminar muestra estudios, tanto a favor como en contra, de la posibilidad de explotar los datos históricos para alcanzar rendimientos extraordinarios en dicho mercado. Esta tesis ha querido contestar la pregunta: ¿No se encuentra información en los datos históricos por qué no existe o las técnicas utilizadas no son apropiadas para ello?

Se ha partido de la hipótesis siguiente: es posible que, para momentos puntuales concretos y para acciones específicas, sea posible a partir de los datos históricos (pretratados por medio de técnicas de análisis técnico y/o estadístico básico), predecir con cierto grado de probabilidad el futuro inmediato de las cotizaciones. Para ello, las técnicas de aprendizaje automático encargadas de extraer dicha información deben poder enfrentar con éxito ciertos factores de dificultad presentes en dichos datos: un ratio muy pequeño de información *versus* ruido, escasez de patrones, alta dimensionalidad y desbalanceo de clases (los eventos más interesantes suelen ser poco frecuentes).

Como **primer objetivo** de esta tesis, se han querido diseñar técnicas, tanto de selección de atributos como de clasificación, robustas a estos factores de dificultad. Para ello, se han desarrollado cuatro variantes de un método de selección de atributos: LIA, LIA2, LIA3 y LIAR (para problemas de regresión). Todas utilizan funciones de evaluación basadas en fundamentos similares pero implementan diferentes métodos de búsqueda de soluciones. Además, se ha desarrollado un método de clasificación, CLIA, especializado en problemas de Tipo-I, también basado en las mismas funciones de evaluación.

Las funciones de evaluación utilizadas giran todas entorno a la idea de, antes de evaluar la utilidad de una *solucion* (región del espacio con una especial densidad de patrones de la clase objetivo), estimar cuantos patrones de cada clase pueden explicarse simplemente por azar. Así, es posible estimar la fiabilidad de una solución para un grado de confianza dada.

Cada técnica utiliza diferente estrategia de búsqueda de soluciones: LIA implementa una búsqueda exhaustiva. LIA2 hace uso de una aproximación *greedy*. Y, LIA3 se basa en una estructura en árbol de decisión. Cada una tiene sus ventajas y desventajas.

Dado que en muchos problemas convencionales de selección de atributos, tanto de clasificación como de regresión, pueden estar presentes algunos de los factores descritos anteriormente, el **segundo objetivo** que se ha planteado en esta tesis es estudiar la utilidad genérica de LIA, LIA2, LIA3 y LIAR, sobre problemas convencionales. No se ha experimentado con CLIA en estos problemas dado que está especializada en problemas Tipo-I.

A lo largo del capítulo 4, **Validación Experimental en problemas Tipo-I**, se ha estudiado si las diferentes variantes de LIA (LIA, LIA2, LIA3) son capaces de enfrentar problemas donde exista poca información y ésta se encuentre semioculta entre un gran conjunto de atributos irrelevantes o ruidosos. Para caracterizar la capacidad de cada técnica de seleccionar los atributos relevantes en escenarios con grados diferentes de dificultad, número de patrones, ruido y número de atributos variables se ha creado un conjunto de 8160 problemas. Dado que los atributos relevantes en estos problemas son conocidos *a-priori* es posible evaluar el desempeño de forma directa.

En cuanto a la comparación entre las diferentes variantes de LIA, podemos resumir que cada variante tiene sus puntos fuertes y sus puntos débiles: LIA por su carácter exhaustivo y gran redundancia, obtiene excelentes resultados. Por contra, escala (en cuanto a tiempo de proceso) con más dificultad. LIA2, es una implementación de compromiso, razonablemente rápida, sencilla y ligera pero obtiene, en general, peores resultados que LIA. Por último, LIA3 presenta un comportamiento intermedio entre ambas variantes tanto en resultados como en tiempo de proceso.

En cualquier caso, todas las variantes de LIA han mostrado un comportamiento prometedor sobre dicho conjunto de datos, comparadas con CFS-Subset, ReliefF y Chi-Squared, a las que han superado en el objetivo de encontrar los atributos más relevantes entre un conjunto de atributos ruidosos. Esta superioridad se ha evidenciado

en mayor medida en los escenarios de mayor dificultad, menos patrones y superior presencia de ruido. En el escenario llamado Difícil, LIA ha obtenido resultados mucho mejores que LIA2 y LIA3. Respecto a las técnicas utilizadas para la comparación, Chi-Squared ha sido la más eficaz en la práctica totalidad de los escenarios frente a ReliefF y `CFS-Subset`.

Dado que los escenarios de mayor dificultad son los que, se piensa, se parecen más al mercado de valores, se ha decidido plantear la segunda fase experimental, predicción en el mercado de valores, comparando la utilidad de LIA (como técnica de selección de atributos) y CLIA (como técnica de clasificación) frente a Chi-Squared y Random Forest como técnicas competidoras. En las pruebas sintéticas se ha visto que Chi-Squared es una buena elección como técnica de selección de atributos en entornos ruidosos. Por su parte, Random Forest, según la bibliografía, es una técnica de clasificación adecuada para este tipo de problemas.

Para estos experimentos, se ha creado una extensa base de problemas reales de bolsa: 117 207 problemas. Cada uno de los cuales consta de 300 patrones y 522 atributos. La valoración se ha realizado en base a la diferencia entre *precision* en la clase C_1, utilizando cada clasificador, frente a la *frecuencia de la clase* C_1 en el mercado. También se han valorado los resultados desde un punto de vista de la rentabilidad económica, comparando la utilidad de cada técnica frente a estrategias de inversión pasivas.

En una primera fase experimental, se ha añadido un atributo extra a los datos de entrada. Dicho atributo se ha generado artificialmente de forma que contiene información relevante para predecir la clase de salida. Se han realizado tres experimentos, variando cada vez la información aportada por los atributos artificiales. Específicamente se ha experimentado con atributos que permiten obtener una precisión máxima en la predicción de la clase C_1 del 55 %, 60 % y 65 %. De este primer bloque experimental se puede concluir que ambas combinaciones de técnicas (LIA/CLIA y Chi-Squared/Random Forest) son capaces de extraer información útil de los atributos informados presentes en el conjunto de datos. Para los tres escenarios planteados LIA/CLIA ha obtenido resultados superiores frente a Chi-Squared/Random Forest.

Por último, se ha experimentado con datos exclusivamente reales, sin información artificial añadida. En éstos, LIA/CLIA ha obtenido diariamente una diferencia positiva respecto a la *frecuencia diaria de la clase* C_1. Chi-Squared/Random Forest también ha obtenido diferencias positivas pero, para la mayor parte de los días, dichas diferencias

han estado por debajo de LIA/CLIA. Es destacable que LIA/CLIA se ha mantenido para la práctica totalidad del experimento en valores positivos. Para ambas técnicas, puede apreciarse como el rendimiento del sistema parece decrecer en el tiempo. Y, es de destacar que la mejora obtenida lo ha sido para muchas empresas, no concentrándose de forma especial en ninguna.

Comparando los resultados obtenidos por cada técnica y observando que la *precision*, con datos reales están para LIA/CLIA muy cercanos a los obtenidos en el caso de añadir artificialmente un atributo que permita alcanzar un *precision* del 55 %, podemos deducir que: la información presente en los datos de entrada (indicadores técnicos sobre datos históricos de bolsa) parece equivaler a la incorporada al escenario Info55. Dado que la *frecuencia global de* C_1 es del 47 %, podemos estimar que LIA/CLIA es capaz de obtener una diferencia positiva del 55 %-47 %=8 %. De forma similar, se puede calcular que la diferencia positiva para Chi-Squared/Random Forest es del 4 %.

Se ha querido verificar también, si existen unos pocos atributos que permitan alcanzar dicha *precision* o si, por el contrario, la información está repartida entre muchos de los atributos utilizados como entrada. Los resultados muestran como los atributos potencialmente informados (atributos que han dado lugar a predicciones correctas), están muy repartidos entre el total de atributos. No obstante, algunos de los indicadores sencillos, calculados a partir del estado global del mercado y no específicos de empresas concretas, parecen ser los que han concentrado mayor utilidad informativa.

Desde un punto de vista del rendimiento económico, frente a otras estrategias pasivas, LIA/CLIA obtiene rentabilidades superiores al resto de técnicas, siempre y cuando se considere un escenario de bajas comisiones.

Vistos en conjunto, estos resultados parecen confirmar que la metodología utilizada, la generación de los patrones y el uso de LIA/CLIA para analizarlos, permite obtener de manera estable en el tiempo, rendimientos superiores a los esperables por azar.

Así, a la pregunta planteada al inicio de esta tesis: *¿Existe información en los datos históricos de bolsa?* se podría contestar, con cierto grado de confianza: sí, los resultados parecen compatibles con la existencia de una pequeña, pero no nula, cantidad de información aprovechable en los datos históricos. *¿Sería rentable un sistema automático que de forma sistemática haga uso de esta metodología para superar de forma recurrente al mercado?* En este caso la respuesta es más compleja. Probablemente sí pueda explotar-

se esta información para obtener mejores resultados económicos, pero el rendimiento final está muy condicionado por las comisiones que se apliquen en cada caso.

Por último, dado que el rendimiento del sistema ha decrecido en el tiempo, se puede apuntar la idea de que quizás los actores del mercado, cada vez más automatizados, estén reduciendo (por hacer uso de ella) la información utilizable. Si esto fuera cierto, podríamos pensar que la automatización del análisis de datos, en tiempo real, podría estar convirtiendo la Bolsa en el mercado perfecto, donde el precio de las acciones refleja instantáneamente todo el conocimiento que en cada momento tenemos sobre el mismo.

En el capítulo 5, **Validación experimental en problemas de clasificación y regresión**, se han presentado los resultados obtenidos con los algoritmos de selección para problemas de clasificación y regresión.

Para los problemas de clasificación se han empleado diferentes conjuntos de problemas reales, obtenidos de repositorios públicos: UCI y KEEL. Se ha estudiado el rendimiento de cada técnica de selección de atributos: LIA, LIA2, LIA3, CFS-Subset, ReliefF y Chi-Squared en combinación con los clasificadores: J48, K-NN y MLP.

Todas las variantes de LIA han obtenido resultados competitivos frente a las otras técnicas de selección: CFS-Subset, ReliefF y Chi-Squared independientemente del clasificador que se use a continuación: J48, KNN ó MLP.

Los promedios normalizados de G-mean, obtenidos con cada técnica, han mostrado que todas las variantes de LIA resultan útiles sea cual sea el porcentaje de atributos y la técnica de clasificación que se quiera utilizar.

En cuando a la comparación de resultados por medio del test de significancia Wilcoxon se ha evidenciado la superioridad de las variantes de LIA, si bien, de forma no tan contundente como en el caso de los promedios.

Para los experimentos de regresión, los datos se han obtenido de KEEL. La técnicas utilizadas en este caso han sido: LIAR (haciendo uso de LIA, LIA2 y LIA3), CFS-Subset y ReliefF combinadas con SMOreg. También, se ha experimentado con un problema de regresión y muchos atributos (1200), aplicable a predicción de producción fotovoltaica. Para este segundo bloque se ha experimentado con LIAR, ReliefF y Correlación Lineal combinadas con GBM como técnica de regresión.

En problemas de regresión, los resultados han mostrado que LIAR mejora el *error medio* frente a `CFS-Subset`, ReliefF y Correlación Lineal. Además, ha podido apreciarse como LIA y LIA2, han obtenido mejores resultados según aumenta *depth*. Resultados

concordantes con los obtenidos en el experimento de predicción fotovoltaica, con un alto número de atributos. En este último, LIA2 ha obtenido unos resultados bastante mejores que el resto.

Tanto para los problemas de clasificación como para los de regresión y para las tres variantes de LIA, se ha podido observar la mejora de los resultados según aumenta la profundidad de exploración (parámetro *depth*). Esto aporta cierto grado de flexibilidad a cada técnica: dependiendo del tiempo total disponible para realizar los cálculos, las diferentes variantes de LIA se pueden parametrizar con $depth = 1$ para obtener un resultado rápido (y, según los experimentos realizados, competitivo) o utilizar $depth = 2$ ó $depth = 3$ para intentar obtener un resultado aún mejor, a costa de un mayor tiempo de proceso.

Como conclusión final a este segundo bloque experimental, vemos que todas las variantes de LIA han mostrado cierta superioridad sobre CFS-Subset, ReliefF y Chi-Squared, en prácticamente todos los escenarios y para todas las técnicas de clasificación (J48, KNN y MLP) y regresión (SMOreg y GBR) utilizadas, especialmente para $depth > 1$.

Por ello, se puede concluir con que LIA, LIA2 y LIA3 son de aplicación en problemas de selección de atributos para clasificación y LIAR lo es en problemas de regresión, en combinación con cualquiera de las anteriores.

Atendiendo a la estabilidad mostrada por LIA3 que ha conseguido buenos resultados para todos los escenarios de clasificación y regresión, se considera la técnica de elección, para este tipo de problemas, de entre las tres variantes desarrolladas.

6.1. Principales aportaciones

Fruto del presente trabajo ha sido el **diseño de nuevas técnicas** de Aprendizaje Automático:

- LIA: Selección de Atributos por Análisis Local de la Fiabilidad.
- LIA2: Implementación *greedy* de LIA.
- LIA3: Implementación de LIA como árbol de decisión.
- LIAR: Implementación de LIA aplicable a problemas de regresión.

- CLIA: Clasificación por Análisis Local de la Fiabilidad.

A nivel conceptual:

- Se han estudiado las características presentes en ámbitos de problemas como el mercado de valores y similares, y su impacto en las técnicas de aprendizaje.
- Se ha detallado el impacto de las características del problema y, concretamente, la monoticidad, en los métodos de búsqueda de soluciones. Se han mostrado diferentes aproximaciones a dicho problema, tanto exhaustivas como *greedy*.
- Se han estudiado diferentes funciones de valoración y cuantificación tanto de la información aportada por soluciones parciales como de la fiabilidad de éstas.
- Se han aportado experimentalmente evidencias de la existencia de una pequeña, pero no nula, cantidad de información utilizable en los datos históricos del mercado de valores.
- Se ha explorado el uso de las técnicas de selección de atributos desarrolladas en dominios de clasificación y regresión, resultando ser técnicas competitivas.

Publicaciones:

- (Martín et al., 2017): *A filter attribute selection method based on local reliable information. Applied Intelligence.* DOI:10.1007/s10489-017-0959-3. `http://rdcu.be/txTD`. JCR (del 2016): Q2.
- (Martín et al., 2016): *Machine learning techniques for daily solar energy prediction and interpolation using numerical weather models. Concurrency and Computation: Practice and Experience. Wiley Online Library.*
  JCR (del 2016): Q3.
- (Aler et al., 2015): *A study of machine learning techniques for daily solar energy forecasting using numerical weather models. Intelligent Distributed Computing VIII. Springer.*

6.2. Futuros trabajos

La técnica de clasificación CLIA, haciendo uso de conceptos similares a LIA, implementa una búsqueda exhaustiva de soluciones. Dicha estrategia resulta en altas tasas de éxito a costa de tiempo de proceso. Estas características hacen que CLIA pueda aplicarse ventajosamente en problemas con pocos atributos, habitualmente menos de 200, donde exista una gran cantidad de ruido y/o atributos irrelevantes. Sin embargo, si el número de atributos es mayor, el tiempo de proceso consumido por la fase I, fase de selección de atributos, puede aumentar demasiado. La fase II.I, fase combinatoria, dependiendo de cómo se parametrice, también puede sufrir problemas de escalabilidad si el número de atributos seleccionados en la fase I superara unas pocas decenas.

LIA2 y LIA3, utilizando aproximaciones *greedy* han demostrado un buen rendimiento en problemas convencionales. Se apunta, como tema interesante, desarrollar una **variante *greedy* de CLIA**. *A grosso modo*, dicha versión podría desarrollarse sustituyendo la fase I, selección de atributos, por LIA2, la implementación *greedy* de LIA. A su vez, la fase II.I podría eliminarse, de forma que sólo se utilizara como clasificador la actual fase II.II, fase incremental. Dicho desarrollo, podría permitir utilizar CLIA en problemas de varios miles de atributos.

Otra posible línea de investigación sería intentar obtener una **nueva técnica de clasificación a partir de LIA3**. En la implementación actual de ésta se construye un árbol de decisión para calcular los atributos más relevantes para la optimización de cierta medida dada. No debería ser demasiado complicado reutilizar dicho árbol, ya construido, para implementar un clasificador basado en el mismo.

El presente trabajo ha estado centrado en el desarrollo de herramientas aplicables al problema de la predicción del mercado de valores, más que a la optimización de los resultados sobre éste. Se apunta, como trabajo futuro, la investigación de **otros eventos a predecir**, utilizando como técnicas de aprendizaje las aquí desarrolladas. Por ejemplo, podría intentarse predecir que valores incrementarán su cotización a lo largo de la semana en curso, o qué valores serán los que más suban porcentualmente el próximo día, mes, trimestre o año...

Anexos

Apéndice A

Tablas para el mercado de valores

Tabla A.1: Indicadores técnicos generados con TA-LIB - (1/2)

Nº Indicador:	Indicador (TA-LIB)	Atributo inicial	Atributo final
1	ADOSC	1	5
2	ADX	6	10
3	ADXR	11	15
4	APO	16	20
5	AROON	21	25
6	AROONOSC	26	30
7	ATR	31	35
8	AVGPRICE	36	40
9	BBANDS	41	45
10	BETA	46	50
11	BOP	51	55
12	CDL2CROWS	56	60
13	CDL3STARSINSOUTH	61	65
14	CDL3WHITESOLDIERS	66	70
15	CDLBREAKAWAY	71	75
16	CDLCLOSINGMARUBOZU	76	80
17	CDLCONCEALBABYSWALL	81	85
18	CDLDARKCLOUDCOVER	86	90
19	CDLDOJI	91	95
20	CDLDOJISTAR	96	100
21	CDLDRAGONFLYDOJI	101	105
22	CDLENGULFING	106	110
23	CDLEVENINGDOJISTAR	111	115
24	CDLEVENINGSTAR	116	120
25	CDLGAPSIDESIDEWHITE	121	125
26	CDLGRAVESTONEDOJI	126	130
27	CDLHAMMER	131	135
28	CDLHANGINGMAN	136	140
29	CDLHARAMI	141	145
30	CDLHARAMICROSS	146	150
31	CDLHIGHWAVE	151	155
32	CDLHIKKAKE	156	160
33	CDLHIKKAKEMOD	161	165
34	CDLHOMINGPIGEON	166	170
35	CDLIDENTICAL3CROWS	171	175
36	CDLINNECK	176	180
37	CDLINVERTEDHAMMER	181	185
38	CDLKICKING	186	190
39	CDLKICKINGBYLENGTH	191	195
40	CDLLADDERBOTTOM	196	200

Tabla A.2: Indicadores técnicos generados con TA-LIB - (2/2)

Nº Indicador:	Indicador (TA-LIB)	Atributo inicial	Atributo final
41	CDLLONGLEGGEDDOJI	201	205
42	CDLMARUBOZU	206	210
43	CDLMATCHINGLOW	211	215
44	CDLMATHOLD	216	220
45	CDLMORNINGDOJISTAR	221	225
46	CDLONNECK	226	230
47	CDLPIERCING	231	235
48	CDLRICKSHAWMAN	236	240
49	LINEARREG	241	245
50	LINEARREG_ANGLE	246	250
51	LINEARREG_INTERCEPT	251	255
52	LINEARREG_SLOPE	256	260
53	MA	261	265
54	MACD	266	270
55	MAX	271	275
56	MAXINDEX	276	280
57	MEDPRICE	281	285
58	MFI	286	290
59	MIDPOINT	291	295
60	MINMAX	296	300
61	MINMAXINDEX	301	305
62	MINUS_DI	306	310
63	MINUS_DM	311	315
64	PLUS_DM	316	320
65	ROC	321	325
66	ROCP	326	330
67	ROCR	331	335
68	ROCR100	336	340
69	RSI	341	345
70	SMA	346	350
71	STDDEV	351	355
72	STOCH	356	360
73	STOCHF	361	365
74	STOCHRSI	366	370
75	TRIMA	371	375
76	TYPPRICE	376	380
77	VAR	381	385
78	WCLPRICE	386	390
79	WILLR	391	395
80	WMA	396	400

Tabla A.3: Resultados por empresa - (1/3)

Empresa	Días válidos cotizados	Frecuencia C_1	Días operados (CLIA)	$PrecisionC_1$
AAL	1502	0,51	47	0,43
ABF	1500	0,49	61	0,49
ALLL	208	0,44	11	0,64
ANTO	1502	0,5	45	0,47
AV	1501	0,49	116	0,55
AZN	1501	0,49	45	0,47
BA	1501	0,43	278	0,52
BARC	1501	0,48	61	0,43
BATS	1500	0,5	52	0,40
BG	1501	0,52	90	0,51
BGY	335	0,41	11	0,45
BLND	1501	0,47	58	0,50
BLT	1501	0,51	62	0,53
BP	1501	0,46	216	0,46
BSY	1502	0,46	56	0,41
BT-A	1502	0,42	56	0,50
CBRY	609	0,45	22	0,36
CCL	1502	0,5	99	0,56
CNA	1499	0,41	50	0,40
CNE	1062	0,51	36	0,50
CPG	1501	0,48	48	0,46
CPI	1497	0,46	192	0,51
CWC	646	0,4	108	0,50
DGE	1502	0,5	51	0,57
DMGT	82	0,45	1	1,00
DRX	270	0,47	33	0,39
DXNS	82	0,33	2	0,00
EMG	1208	0,45	43	0,40
ETI	272	0,39	10	0,60
EXPN	1502	0,49	145	0,53
FP	522	0,39	15	0,53
GSK	1501	0,48	46	0,54
HBOS	322	0,41	6	0,33
HIBU	144	0,46	6	0,50
HMSO	1502	0,48	164	0,50
HOME	646	0,41	104	0,54
HSBA	1500	0,48	43	0,49
IAG	1500	0,45	223	0,58
IAP	1271	0,46	41	0,49
ICI	56	0,29	11	0,55

Tabla A.4: Resultados por empresa - (2/3)

Empresa	Días válidos cotizados	Frecuencia C_1	Días operados (CLIA)	$PrecisionC_1$
IHG	1501	0,51	57	0,49
III	957	0,45	116	0,56
IMT	1501	0,51	65	0,42
INTU	1396	0,47	151	0,60
IPR	1184	0,46	140	0,58
ITV	274	0,34	6	0,50
JMAT	1501	0,5	64	0,47
KAZ	1396	0,52	40	0,53
KGF	1502	0,41	276	0,53
LAND	1502	0,48	138	0,57
LGEN	1502	0,37	295	0,48
LLOY	1502	0,36	246	0,46
LMI	1020	0,49	80	0,60
MAB	82	0,38	4	0,50
MKS	1501	0,45	59	0,54
MRW	1501	0,42	44	0,55
NG	1499	0,47	218	0,58
NRK	82	0,39	15	0,53
NXT	1502	0,49	153	0,54
OML	1501	0,42	43	0,33
PRU	1502	0,49	128	0,62
PSN	209	0,44	33	0,55
PSON	1502	0,48	60	0,57
PUB	82	0,41	2	0,50
RB	1501	0,5	58	0,43
RBS	1502	0,37	41	0,29
RDSA	1500	0,48	43	0,44
RDSB	1502	0,49	57	0,51
REL	1502	0,47	49	0,39
REX	1501	0,47	43	0,51
RIO	1500	0,52	51	0,65
RR	1502	0,48	143	0,52
RSA	1501	0,35	276	0,49
RSL	145	0,41	8	0,38
RTO	143	0,4	5	1,00
SAB	1502	0,48	52	0,33
SBRY	1500	0,4	289	0,51
SCTN	143	0,41	3	0,00
SDR	1502	0,5	56	0,50
SDRC	1207	0,5	37	0,54

Tabla A.5: Resultados por empresa - (3/3)

Empresa	Días válidos cotizados	Frecuencia C_1	Días operados (CLIA)	$PrecisionC_1$
SGE	1501	0,42	271	0,53
SHP	1502	0,49	117	0,65
SL	1501	0,43	286	0,53
SMIN	1501	0,49	142	0,59
SN	1478	0,45	55	0,47
SSE	1502	0,49	136	0,54
STAN	1500	0,48	47	0,55
SVT	1501	0,5	71	0,56
TATE	205	0,43	15	0,33
TRIL	487	0,4	24	0,38
TSCO	1502	0,43	265	0,53
ULVR	1502	0,47	40	0,30
UU	1502	0,47	47	0,47
VED	1498	0,5	43	0,51
VOD	1501	0,38	336	0,49
WOS	1436	0,48	146	0,53
WPP	1502	0,49	144	0,60
WTB	1435	0,5	96	0,56
XTA	1395	0,52	49	0,57
TOTAL	**117 207**	**0,47**	**8742**	**0,52**

Referencias Bibliográficas

Aguilar-Ruiz, J. and Diaz-Diaz, N. (2005). Selección de atributos relevantes basada en bootstrapping. 29

Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. *Machine Learning: ECML 2004*, pages 39–50. 25

Akerlof, G. (1995). *The market for "lemons": Quality uncertainty and the market mechanism*. Springer. 14

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17. 156, 170

Aler, R., Martín, R., Valls, J. M., and Galván, I. M. (2015). A study of machine learning techniques for daily solar energy forecasting using numerical weather models. In *Intelligent Distributed Computing VIII*, pages 269–278. Springer. 191

Alexander, S. S. (1961). Price movements in speculative markets: Trends or random walks. *Industrial Management Review (pre-1986)*, 2(2):7. 11, 22

Alkhatib, K., Najadat, H., Hmeidi, I., and Shatnawi, M. K. A. (2013). Stock price prediction using k-nearest neighbor (knn) algorithm. *International Journal of Business, Humanities and Technology*, 3(3):32–44. 19

Almuallim, H. and Dietterich, T. G. (1991). Learning with many irrelevant features. In *AAAI*, volume 91, pages 547–552. 31

Andrew, L. (1997). A non-random walk down wall street. In *Proceedings of symposia in pure mathematics*, volume 60, pages 149–183. 15

Asuncion, A. and Newman, D. (2007). Uci machine learning repository. 156

Bachelier, L. (1900). Teoria de la especulación. 10

Ballings, M., Van den Poel, D., Hespeels, N., and Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056. 20

Balsara, N., Carlson, K., and Rao, N. V. (1996). Unsystematic futures profits with technical trading rules: A case for flexibility. *Journal of Financial and Strategic Decisions*, 9(1):57–66. 15

Balsara, N. J., Chen, G., and Zheng, L. (2007). The chinese stock market: An examination of the random walk model and technical trading rules. *Quarterly Journal of Business and Economics*, pages 43–63. 23

Barandela, R., Valdovinos, R., Sanchez, J., and Ferri, F. (2004). The imbalanced training sample problem: Under or over sampling? *Structural, Syntactic, and Statistical Pattern Recognition*, pages 806–814. 25

Bastos, J. A. and Caiado, J. (2011). Recurrence quantification analysis of global stock markets. *Physica A: Statistical Mechanics and its Applications*, 390(7). 16

Batchelor, R. and Ramyar, R. (2006). Magic numbers in the dow. 22

Bellman, R. (1961). Adaptive control processes: A guided tour. 26

Ben-Bassat, M. (1982). Pattern recognition and reduction of dimensionality. *Handbook of Statistics*, 2:773–910. 30

Bennasar, M., Hicks, Y., and Setchi, R. (2015). Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532. 30

REFERENCIAS BIBLIOGRÁFICAS

Bermejo, P., Gámez, J. A., and Puerta, J. M. (2011). A grasp algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets. *Pattern Recognition Letters*, 32(5):701–711. 32

Bhattacharya, S. and Kumar, K. (2006). A computational exploration of the efficacy of fibonacci sequences in technical analysis and trading. 22

Bigalow, S. W. (2011). *Profitable candlestick trading: pinpointing market opportunities to maximize profits*, volume 500. John Wiley & Sons. 22

Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. 27

Bollerslev, T., Chou, R. Y., and Kroner, K. F. (1992). Arch modeling in finance: A review of the theory and empirical evidence. *Journal of econometrics*, 52(1-2):5–59. 13

Bollerslev, T., Engle, R. F., and Nelson, D. B. (1994). Arch models. *Handbook of econometrics*, 4:2959–3038. 13

Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2013). A review of feature selection methods on synthetic data. *Knowledge and information systems*, 34(3):483–519. 32, 33

Bondt, W. F. and Thaler, R. (1985). Does the stock market overreact? *The Journal of finance*, 40(3):793–805. 14

Booth, A., Gerding, E., and Mcgroarty, F. (2014). Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, 41(8):3651–3661. 20

Brassard, G. and Bratley, P. (1996). *Fundamentals of algorithmics*, volume 33. Prentice Hall Englewood Cliffs. 30

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140. 38

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 24, 38

Brock, W., Lakonishok, J., and LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47(5):1731–1764. 22

Brown, G., Pocock, A., Zhao, M.-J., and Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13(Jan):27–66. 32

Brown, S. J., Goetzmann, W. N., and Kumar, A. (1998). The dow theory: William peter hamilton's track record reconsidered. *The Journal of finance*, 53(4):1311–1333. 22

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167. 39

Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the tenth international conference on machine learning*, volume 25, page 32. Amherst, MA. 30

Cawley, G. C. and Talbot, N. L. (2006). Gene selection in cancer classification using sparse logistic regression with bayesian regularization. *Bioinformatics*, 22(19):2348–2355. 32

Chan, K., Hameed, A., and Tong, W. (2000). Profitability of momentum stragegies in the international equity markets. *Journal of Financial and Quantitative Analysis*, 35(02):153–172. 23

Chan, L. K., Jegadeesh, N., and Lakonishok, J. (1996). Momentum strategies. *The Journal of Finance*, 51(5):1681–1713. 15

Chaudhuri, K. and Wu, Y. (2003). Random walk versus breaking trend in stock prices: Evidence from emerging markets. *Journal of Banking & Finance*, 27(4):575–592. 22

Chen, J., Hong, H., and Stein, J. C. (2001). Forecasting crashes: Trading volume, past returns, and conditional skewness in stock prices. *Journal of Financial Economics*, 61(3):345–381. 23

Chen, J., Hong, H., and Stein, J. C. (2002). Breadth of ownership and stock returns. *Journal of financial Economics*, 66(2):171–205. 23

Chen, S.-H. and Navet, N. (2007). Failure of genetic-programming induced trading strategies: Distinguishing between efficient markets and inefficient algorithms. In *Computational Intelligence in Economics and Finance*, pages 169–182. Springer. 19

Chen, X.-w. (2003). An improved branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 24(12):1925–1933. 29

REFERENCIAS BIBLIOGRÁFICAS

Chopra, N., Lakonishok, J., and Ritter, J. R. (1992). Measuring abnormal performance: do stocks overreact? *Journal of financial Economics*, 31(2):235–268. 15

Connolly, R. and Stivers, C. (2003). Momentum and reversals in equity-index returns during periods of abnormal turnover and return dispersion. *The Journal of Finance*, 58(4):1521–1556. 23

Cootner, P. H. (1962). Stock prices: Ramdom vs. systematic changes. *Industrial Management Review (pre-1986)*, 3(2):24. 14

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. 39

Covel, M. (2004). *Trend following: how great traders make millions in up or down markets*. FT Press. 22

Cover, T. and Van Campenhout, J. (1977). On the possible orderings in the measurement selection problem. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(9):657–661. 29, 61

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*. 38

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 74–81. 28, 29

Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156. 29

Davies, S. and Russell, S. (1994). Np-completeness of searches for smallest possible feature sets. In *Proceedings of the AAAI Fall'94 Symposium on Relevance, New Orleans*. 29, 61

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30. 155

Doak, J. (1992). Cse-92-18-an evaluation of feature selection methodsand their application to computer security. *UC Davis Dept of Computer Science tech reports*. 29, 30

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM. 26

Domingos, P. M. (1997). Why does bagging work? a bayesian account and its implications. In *KDD*, pages 155–158. Citeseer. 28

Duarte, Juan Benjamín, D. and Perez-Iñigo, J. M. M. (2013). La eficiencia de los mercados de valores: una revisión. *Análisis financiero*. 17

Dy, J. and Brodley, C. (2004). Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:845–889. 28

Eitrich, T., Kless, A., Druska, C., Meyer, W., and Grotendorst, J. (2007). Classification of highly unbalanced cyp450 data of drugs using cost sensitive machine learning techniques. *Journal of chemical information and modeling*, 47(1):92–103. 25

Exuberance, I. (2000). Princeton (nj). 16

Fama, E. F. (1963). Mandelbrot and the stable paretian hypothesis. *The journal of business*, 36(4):420–429. 11

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, 38(1):34–105. 10, 11

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *TheJournal of Finance*. 10

Fama, E. F. (1991). Efficient capital markets: Ii. *The journal of finance*, 46(5):1575–1617. 10

Fama, E. F. (1998). Market efficiency, long-term returns, and behavioral finance. *Journal of financial economics*, 49(3):283–306. 12

Fama, E. F., Fisher, L., Jensen, M. C., and Roll, R. (1969). The adjustment of stock prices to new information. *International economic review*, 10(1):1–21. 11

Fama, E. F. and French, K. R. (1988). Permanent and temporary components of stock prices. *Journal of political Economy*, 96(2):246–273. 15

Fang, Y. and Xu, D. (2003). The predictability of asset returns: an approach combining technical analysis and time series forecasts. *International Journal of Forecasting*, 19(3):369–385. 22

Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316. 2

REFERENCIAS BIBLIOGRÁFICAS

Fernández, A., García, S., del Jesus, M. J., and Herrera, F. (2008). A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems*, 159(18):2378–2398. 112, 156

Fernández Rodríguez, F., Sosvilla Rivero, S., and Andrada Félix, J. (1999). Technical analysis in the madrid stock exchange. 22

Ferreira, A. J. and Figueiredo, M. A. (2012). Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters*, 33(13):1794–1804. 31, 32

Fischer, R. and Fischer, J. (2003). *Candlesticks, Fibonacci, and chart pattern trading tools: a synergistic strategy to enhance profits and reduce risk*, volume 209. John Wiley & Sons. 22

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305. 27

Foroutan, I. and Sklansky, J. (1987). Feature selection for automatic classification of non-gaussian data. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(2):187–198. 29

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin. 26

Friedman, J. H. (1987). Greedy function approximation: A gradient boosting machine. *mh (xa m)*, 1000:0. 23

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232. 39

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378. 23, 39

Fukunaga, K. and Hayes, R. (1989). Effects of sample size in classifier design. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(8):873–885. 26

Fung, W. and Hsieh, D. A. (2001). The risk in hedge fund strategies: Theory and evidence from trend followers. *Review of Financial studies*, 14(2):313–341. 22

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484. 39

Gao, S., Lee, C.-H., and Lim, J. H. (2006). An ensemble classifier learning approach to roc optimization. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 679–682. IEEE. 25

Gartley, H. M. (1935). *Profits in the stock market*. Health Research Books. 22

Gil, U., Mario, J., and Ulloa Villegas, I. M. (2011). Revisando la hipótesis de los mercados eficientes: nuevos datos, nuevas crisis y nuevas estimaciones. *Cuadernos de Economía*, 30(55):127–154. 16

Graham, B. and Dodd, D. L. (1934). *Security Analysis: Principles and Technique*. McGraw-Hill. 9

Granger, C. W. and Morgenstern, O. (1963). Spectral analysis of new york stock market prices. *Kyklos*, 16(1):1–27. 14

Granville, J. E. (1963). *New key to stock market profits*. Prentice-Hall. 23

Grossman, S. J. and Stiglitz, J. E. (1980). On the impossibility of informationally efficient markets. *The American economic review*, 70(3):393–408. 14

Grundy, B. D. and Martin, J. S. M. (2001). Understanding the nature of the risks and the source of the rewards to momentum investing. *Review of Financial studies*, 14(1):29–78. 23

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182. 27, 28, 31

Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A. (2008). *Feature extraction: foundations and applications*, volume 207. Springer. 31

Hall, M. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato. 28, 30, 31, 32, 34

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18. 24, 34

Hall, M. A. and Smith, L. A. (1997). Feature subset selection: a correlation based filter approach. pages 855–858. 34

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36. 24

Hastie, T., Tibshirani, R., and Friedman, J. (2003). The elements of statistical learning: Data mining, inference, and prediction. 26, 38

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learning 2nd edition. 31

Haugen, R. A. (1995). *The new finance: the case against efficient markets*. Prentice Hall. 15

Ho, T. K. (1995). Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE. 38

Hua, J., Tembe, W., and Dougherty, E. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–424. 31

Huang, C.-J., Yang, D.-X., and Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, 34:2870–2878. 19

Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522. 18

Huddart, S. J., Lang, M. H., and Yetman, M. (2005). Psychological factors, stock price paths, and trading volume. In *AFA 2006 Boston Meetings Paper*. 23

Hughes, G. P. (1968). On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63. 26

Hughes, T. J. (1995). Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer methods in applied mechanics and engineering*, 127(1):387–401. 26

Hwang, J., Lay, S., and Lippman, A. (1994). Nonparametric multivariate density estimation: a comparative study. *Signal Processing, IEEE Transactions on*, 42(10):2795–2810. 26

Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17. 37

Ichino, M. and Sklansky, J. (1984a). Feature selection for linear classifier. In *Proceedings of the Seventh International Conference on Pattern Recognition*, volume 1, pages 124–127. 29

Ichino, M. and Sklansky, J. (1984b). Optimum feature selection by zero-one integer programming. *IEEE Transactions on Systems, Man, and Cybernetics*, (5):737–746. 29

Inza, I., Larrañaga, P., Blanco, R., and Cerrolaza, A. J. (2004). Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial intelligence in medicine*, 31(2):91–103. 28, 31

Jackson, M. O. (1991). Equilibrium, price formation, and the value of private information. *Review of Financial Studies*, 4(1):1–16. 12

Jain, A. and Waller, W. (1978). On the optimal number of features in the classification of multivariate gaussian data. *Pattern recognition*, 10(5-6):365–374. 26

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449. 2

Jegadeesh, N. and Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance*, 48(1):65–91. 15

Jimenez, L. and Landgrebe, D. (1998). Supervised classification in high-dimensional space: Geometrical, statistical, and asymptotical properties of multivariate data. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(1):39–54. 26

Kaminsky, G. L. and Schmukler, S. L. (1999). What triggers market jitters?: A chronicle of the asian crisis. *Journal of international money and Finance*, 18(4):537–560. 16

Karegowda, A. G., Manjunath, A., and Jayaram, M. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277. 29

REFERENCIAS BIBLIOGRÁFICAS

Kaufman, P. J. (2013). *Trading Systems and Methods*, volume 591. John Wiley & Sons. 10

Kavajecz, K. A. and Odders-White, E. R. (2004). Technical analysis and liquidity provision. *Review of Financial Studies*, 17(4):1043–1071. 22

Khushaba, R. N., Al-Ani, A., and Al-Jumaily, A. (2011). Feature subset selection using differential evolution and a statistical repair mechanism. *Expert Systems with Applications*, 38(9):11515–11526. 30, 32

Kim, Y., Street, W. N., and Menczer, F. (2000). Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 365–369. ACM. 28

Kimoto, T., Asakawa, K., Yoda, M., and Takeoka, M. (1990). Stock market prediction system with modular neural networks. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 1–6. IEEE. 18

Kira, K. and Rendell, L. A. (1992a). The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134. 34

Kira, K. and Rendell, L. A. (1992b). A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. 28, 30, 32, 34

Kohavi, R. and John, G. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324. 28, 29

Kohavi, R. and John, G. H. (1998). The wrapper approach. In *Feature extraction, construction and selection*, pages 33–50. Springer. 28

Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. In *Machine Learning: ECML-94*. Springer. 34

Kononenko, I., Šimec, E., and Robnik-Šikonja, M. (1997). Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence*, 7(1):39–55. 34

Krishnapuram, B., Carin, L., Figueiredo, M., and Hartemink, A. (2005). Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE. Trans. Pattern. Anal. Mach. Intell.* 32

Ładyżyński, P., Żbikowski, K., and Grzegorzewski, P. (2013). Stock trading with random forests, trend detection tests and force index volume indicators. In *International Conference on Artificial Intelligence and Soft Computing*, pages 441–452. Springer. 20

Lakonishok, J., Shleifer, A., and Vishny, R. W. (1994). Contrarian investment, extrapolation, and risk. *The journal of finance*, 49(5):1541–1578. 15

Lee, C. and Swaminathan, B. (2000). Price momentum and trading volume. *the Journal of Finance*, 55(5):2017–2069. 23

Lee, C.-C., Lee, J.-D., and Lee, C.-C. (2010). Stock prices and the efficient market hypothesis: Evidence from a panel stationary test with structural breaks. *Japan and the world economy*, 22(1):49–58. 16

Lento, C. and Gradojevic, N. (2011). The profitability of technical trading rules: A combined signal approach. *Journal of Applied Business Research (JABR)*, 23(1). 16

Liang, J., Yang, S., and Winstanley, A. (2008). Invariant optimal feature selection: A distance discriminant and feature ranking based solution. *Pattern Recognition*, 41(5):1429–1439. 30, 32

Liu, H. and Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media. 29, 30, 31

Liu, H., Motoda, H., and Yu, L. (2002). Feature selection with selective sampling. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 395–402. 30

Liu, H. and Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. In *In Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, page 388. IEEE. 28, 30, 35

Liu, H., Setiono, R., et al. (1996). A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer. 30, 31

Liu, H., Sun, J., Liu, L., and Zhang, H. (2009). Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339. 27

Liu, H., Wu, X., and Zhang, S. (2014). A new supervised feature selection method for pattern classification. *Computational Intelligence*, 30(2):342–361. 27

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502. 27, 28, 29

Lorenzo Navarro, J. (2001). *Selección de atributos basado en aprendizaje automático basada en teoría de la información*. PhD thesis, Las Palmas de Gran Canaria. 29

Loughrey, J. and Cunningham, P. (2005). Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In *Research and Development in Intelligent Systems XXI*, pages 33–43. Springer. 28

Malkiel, B. (1992). Eficient market hypothesis. *New Palgrave Dictionary of Money and Finance*. 10

Malkiel, B. G. (1973a). Un paseo aleatorio por wall street. 10, 142

Malkiel, B. G. (1973b). Un paseo aleatorio por wall street. 12

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *The Journal of Economic Perspectives*, 17(1):59–82. 12

Malkiel, B. G. (2005). Reflections on the efficient market hypothesis: 30 years later. *Financial Review*, 40(1):1–9. 12

Malkiel, B. G. and Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417. 10

Manahov, V., Hudson, R., and Gebka, B. (2014). Does high frequency trading affect technical analysis and market efficiency? and if so, how? *Journal of International Financial Markets, Institutions and Money*, 28(C):131–157. 17

Mandelbrot, B. (1966). Forecasts of future prices, unbiased markets, and"martingale"models. *The Journal of Business*, 39(1):242–255. 10, 11

Mandelbrot, B. B. (1963). The variation of certain speculative prices. *The Journal of Business*, 36(4):394–419. 11

Mandic, D. P. and Goh, V. S. L. (2001). Adaptive and learning systems for signal, processing, communications, and control. *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*, pages 325–325. 39

Maragoudakis, M. and Serpanos, D. (2010). Towards stock market data mining using enriched random forests from textual resources and technical indicators. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 278–286. Springer. 19

Marshall, B. R., Cahan, R. H., and Cahan, J. M. (2008a). Can commodity futures be profitably traded with quantitative market timing strategies? *Journal of Banking & Finance*, 32(9):1810–1819. 13

Marshall, B. R., Cahan, R. H., and Cahan, J. M. (2008b). Does intraday technical analysis in the us equity market have value? *Journal of Empirical Finance*, 15(2):199–210. 13

Martín, R., Aler, R., and Galvan, I. (2017). A filter attribute selection method based on local reliable information. *Applied Intelligence*, pages 1–11. doi:10.1007/s10489-017-0959-3. 191

Martín, R., Aler, R., Valls, J., and Galvan, I. (2016). Machine learning techniques for daily solar energy prediction and interpolation using numerical weather models. *Concurrency and Computation: Practice and Experience*. 191

Metcalf, G. E. and Malkiel, B. G. (1994). The wall street journal contests: the experts, the darts, and the efficient market hypothesis. *Applied Financial Economics*, 4(5):371–374. 12

Mizuno, H., Kosaka, M., Yajima, H., and Komoda, N. (1998). Application of neural network to technical analysis of stock market prediction. *Studies in Informatic and control*, 7(3):111–120. 18

Moore, A. B. (1962). *A statistical analysis of common stock prices*. PhD thesis, University of Chicago. 14

Mucciardi, A. and Gose, E. (1971). A comparison of seven techniques for choosing subsets of pattern recognition properties. *Computers, IEEE Transactions on*, 100(9):1023–1031. 30

Naeini, M. P., Taremian, H., and Hashemi, H. B. (2010). Stock market value prediction using neural networks. In *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, pages 132–136. IEEE. 19

Nakariyakul, S. and Casasent, D. P. (2007). Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognition Letters*, 28(12):1415–1427. 29, 32

Narendra, P. M. and Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26(9):917–922. 29

REFERENCIAS BIBLIOGRÁFICAS

Nison, S. (2004). *The Candlestick Course*, volume 163. John Wiley & Sons. 22

Oh, K. J. and Kim, K.-j. (2002). Analyzing stock market tick data using piecewise nonlinear model. *Expert Systems with Applications*, 22:249–255. 16

Ohira, T., Sazuka, N., Marumo, K., Shimizu, T., Takayasu, M., and Takayasu, H. (2002). Predictability of currency market exchange. *Physica A: Statistical Mechanics and its Applications*, 308(1):368–374. 16

Okunev, J. and White, D. (2003). Do momentum-based strategies still work in foreign currency markets? *Journal of Financial and Quantitative Analysis*, 38(02):425–447. 23

Osborne, M. F. (1962). Periodic structure in the brownian motion of stock prices. *Operations Research*, 10(3):345–379. 14

Osborne, M. M. (1959). Brownian motion in the stock market. *Operations research*, 7(2):145–173. 11

Osler, C. L. (2000). Support for resistance: technical analysis and intraday exchange rates. *Economic Policy Review*, 6(2). 22

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42:259–268. 20

Pearson, K. (1905). The problem of the random walk. *Nature*, 72(1865):294. 11

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238. 32

Pietraszek, T. (2007). On the use of roc analysis for the optimization of abstaining classifiers. *Machine Learning*, 68(2):137–169. 25

Poterba, J. M. and Summers, L. H. (1988). Mean reversion in stock prices: Evidence and implications. *Journal of financial economics*, 22(1):27–59. 15

Pudil, P., Novovičová, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125. 30

Qian, B. and Rasheed, K. (2007). Stock market prediction with multiple classifiers. *Applied Intelligence*, 26(1):25–33. 19

Qin, Q., Wang, Q.-G., Li, J., and Ge, S. S. (2013). Linear and nonlinear trading models with gradient boosted random forests and application to singapore stock market. 19

Quinlan, J. (1993). C4. 5: Programs for empirical learning morgan kaufmann. *San Francisco, CA*. 36

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106. 24, 36

Quinlan, J. R. et al. (1979). *Discovering rules by induction from large collections of examples*. Expert systems in the micro electronic age. Edinburgh University Press. 24, 36

Quinlan, J. R. and Rivest, R. L. (1989). Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248. 24, 36

Rangvid, J. (2001). Increasing convergence among european stock markets?: A recursive common stochastic trends analysis. *Economics Letters*, 71(3):383–389. 22

Reboredo, J. C., Matías, J. M., and Garcia-Rubio, R. (2012). Nonlinearity in forecasting of high-frequency stock returns. *Computational Economics*, pages 1–20. 17

Rechenthin, M. and Street, W. N. (2013). Using conditional probability to identify trends in intra-day high-frequency equity pricing. *Physica A: Statistical Mechanics and its Applications*, 392(24):6169–6188. 17

Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *The Journal of Machine Learning Research*, 3:1371–1382. 28

Ridgeway, G., Southworth, M. H., and RUnit, S. (2013). Package 'gbm'. *Viitattu*, 10:2013. 40

Roberts, H. V. (1967). Statistical versus clinical prediction of the stock market. 10

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, page 386. 37

Rothschild, M. and Stiglitz, J. (1976). *Equilibrium in competitive insurance markets: An essay on the economics of imperfect information*. Springer. 14

Royston, J. (1982). Algorithm as 181. *Applied Statistics*, 31(2):176–180. 155

Ruiz, F. E., Pérez, P. S., and Bonev, B. I. (2009). *Information theory in computer vision and pattern recognition*. Springer Science & Business Media. 31

Ruiz, R., Riquelme, J., and Aguilar-Ruiz, J. (2002). Projection-based measure for efficient feature selection. *Journal of Intelligent and Fuzzy Systems-Applic in Engineering and Technology*, 12(3-4):175–184. 31

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document. 37

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):2013. 37

Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517. 27

Samuelson, P. (1965). Proof that properly anticipated prices fluctuate randomly. *Industrial Management Review*. 10

Samuelson, P. A. et al. (1965). *Proof that properly anticipated prices fluctuate randomly*. 11

Sazuka, N. (2006). Analysis of binarized high frequency financial data. *The European Physical Journal B-Condensed Matter and Complex Systems*, 50(1-2):129–131. 16

Schonlau, M. et al. (2005). Boosted regression (boosting): An introductory tutorial and a stata plugin. *Stata Journal*, 5(3):330. 39

Schulenburg, S. and Ross, P. (1999). An evolutionary approach to modelling the behaviours of financial traders. In *Genetic and Evolutionary Computation Conference Late Braking Papers*, pages 245–253. 18

Schulenburg, S. and Ross, P. (2000). Strength and money: An lcs approach to increasing returns. In *Advances in Learning Classifier Systems*, pages 114–137. Springer. 18

Schulenburg, S. and Ross, P. (2001). Explorations in lcs models of stock trading. In *IWLCS*, pages 151–180. Springer. 18

Schwager, J. D. (1996). *Technical analysis*, volume 43. John Wiley & Sons. 10

Schwert, G. W. (2003). Anomalies and market efficiency. *Handbook of the Economics of Finance*, 1:939–974. 12

Scott, D. (1992). *Multivariate density estimation*, volume 139. Wiley Online Library. 26

Seiler, M. J. and Rom, W. (1997). A historical analysis of market efficiency: Do historical returns follow a random walk. *Journal of Financial and Strategic Decisions*, 10(2):49–57. 12

Sewell, M. (2001). Technical analysis. 10

Sewell, M. (2011). History of the efficient market hypothesis. *RN*, 11(04):04. 13

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press. 26, 36

Shevade, S., Keerthi, S., Bhattacharyya, C., and Murthy, K. (1999). Improvements to the smo algorithm for svm regression. 39

Shevade, S. K. and Keerthi, S. S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253. 32

Shiller, R. J. (1980). Do stock prices move too much to be justified by subsequent changes in dividends? 14

Silber, W. L. (1994). Technical trading: when it works and when it doesn't. *The Journal of Derivatives*, 1(3):39–44. 15

Slaughter, R. A. (2002). Beyond the mundane: reconciling breadth and depth in futures enquiry. *Futures*, 34(6):493–507. 23

Smola, A. and Schoelkopf, B. (1998). A tutorial on support vector regression. NeuroCOLT2 Technical Report NC2-TR-1998-030. 39

REFERENCIAS BIBLIOGRÁFICAS

Somol, P., Baesens, B., Pudil, P., and Vanthienen, J. (2005). Filter-versus wrapper-based feature selection for credit scoring. *International Journal of Intelligent Systems*, 20(10):985–999. 28

Spence, M. (1973). Job market signaling. *The quarterly journal of Economics*, pages 355–374. 14

Stiglitz, J. E. (2010). *Freefall: America, free markets, and the sinking of the world economy*. WW Norton & Company. 16

Stracuzzi, D. J. and Utgoff, P. E. (2004). Randomized variable elimination. *Journal of Machine Learning Research*, 5(Oct):1331–1362. 28, 30

Sun, X., Liu, Y., Li, J., Zhu, J., Chen, H., and Liu, X. (2012). Feature evaluation and selection with cooperative game theory. *Pattern recognition*, 45(8):2992–3002. 32

Tanaka-Yamawaki, M. (2003). Stability of markovian structure observed in high frequency foreign exchange data. *Annals of the Institute of Statistical Mathematics*, 55(2):437–446. 16

Taylor, M. P. and Allen, H. (1992). The use of technical analysis in the foreign exchange market. *Journal of international Money and Finance*, 11(3):304–314. 15

Taylor, S. J. and Tari, A. (1989). Further evidence against the efficiency of futures markets. In *A Reappraisal of the Efficiency of Financial Markets*, pages 577–601. Springer. 15

Team, R. C. (2014). R: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria. 2013. 40, 169

Timmermann, A. and Granger, C. W. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, 20(1):15–27. 12

Tóth, B. and Kertész, J. (2006). Increasing market efficiency: Evolution of cross-correlations of stock returns. *Physica A: Statistical Mechanics and its Applications*, 360(2):505–515. 12

Uysal, A. K. (2016). An improved global feature selection scheme for text classification. *Expert Systems with Applications*, 43:82–92. 29

Vafaie, H. and Imam, I. F. (1994). Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, volume 51. 28

Weiss, G. (1995). Learning with rare cases and small disjuncts. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 558–565. MORGAN KAUFMANN PUBLISHERS, INC. 2

Wilcox, C. and Crittenden, E. (2005). Does trend-following work on stocks? *The Technical Analyst*, 14:1–19. 22

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, 1:80–83. 155

Wilson, E. J. and Marashdeh, H. A. (2007). Are co-integrated stock prices consistent with the efficient market hypothesis? *Economic Record*, 83(s1):S87–S93. 13

Witten, I., Frank, E., and Hall, M. (2011). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann. 28, 30, 31, 34, 113, 154, 168

Wong, S. Y. B. and Schulenburg, S. (2007). Portfolio allocation using xcs experts in technical analysis, market conditions and options market. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*, pages 2965–2972. ACM. 18

Wyckoff, R. D. (1910). *Studies in tape reading*. Traders Press. 22

Xing, E. P., Jordan, M. I., Karp, R. M., et al. (2001). Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer. 29, 32

Yao, Y., Wong, S., and Butz, C. (1999). On information-theoretic measures of attribute importance. *Methodologies for Knowledge Discovery and Data Mining*, pages 133–137. 30

Yeh, I.-C. and Hsu, T.-K. (2014). Exploring the dynamic model of the returns from value stocks and growth stocks using time series mining. *Expert Systems with Applications*, 41(17):7730–7743. 17

Yen, G. and Lee, C.-f. (2008). Efficient market hypothesis (emh): past, present and future. *Review of Pacific Basin Financial Markets and Policies*, 11(02):305–329. 13

Yu, L. and Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 856. 29, 31, 32

Yu, L. and Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224. 29

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62. 18

Zhou, Z. and Liu, X. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):63–77. 26

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press. 39