FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

Trabajo de Diploma

Minería de datos para series temporales en Weka y su aplicación en el pronóstico de precipitaciones

Autor: César Soto Valero
Tutores: MSc. Mabel Gonzáles Castellanos
Dra. Yanet Rodríguez Sarabia
Consultante: Dr. Aldo Moya

Santa Clara, 2013

Hago constar que el presente trabajo fue realizado en la Universidad Central Marta Abreu de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del jefe del Seminario

DEDICATORIA Y AGRADECIMIENTOS

A todos…, por todo.

SÍNTESIS

Las series temporales permiten describir una gran variedad de fenómenos que transcurren a lo largo del tiempo. Los modelos que realizan análisis de series temporales usando técnicas de minería de datos son capaces de resolver múltiples problemas, superando las limitaciones de los métodos estadísticos tradicionales. Weka es un poderoso sistema de aprendizaje automatizado, sin embargo, ofrece muy pocas posibilidades para el trabajo con series temporales.

En el presente Trabajo de Diploma se diseña e implementa un paquete para Weka, con herramientas desarrolladas especialmente para la clasificación de series temporales. Dichas herramientas son: una función de distancia basada en DTW, un algoritmo de búsqueda de vecinos más cercanos para kNN y un filtro para la reducción de la numerosidad de series temporales. Además, se aborda el problema del pronóstico de precipitaciones aplicando técnicas de minería de datos a las salidas numéricas del modelo global GFS.

ABSTRACT

Time series can describe a variety of events that take place over time. The model that analyzes time series using data mining techniques is able to solve a lot of problems. This is because time series data minning model overcoming the limitations of traditional statistical methods. Weka is a powerful machine learning system, however, offers very few possibilities for working with time series data.

In this work is designed and implemented a new package for Weka with tools developed especially for the classification of time series. Such tools include a distance function based on DTW, a nearest neighbor search algorithm for kNN and a filter for reducing numerosity. Also, it focuses the issue of applying data mining techniques to rainfall forecast problem using the numerical outputs of GFS global model.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Las series temporales se obtienen mediante la medición de variables a través del tiempo. Resulta difícil imaginar una rama de la ciencia en la que no aparezcan datos que puedan ser considerados como series temporales, por lo que su procedencia abarca los más diversos dominios.

Una serie de tiempo está constituida por observaciones históricas de una o varias variables y por tanto sus valores son irrepetibles. Los datos almacenados en forma de series temporales son susceptibles a contener información valiosa para su dominio de procedencia. De ahí que su utilización se haya enfocado tradicionalmente en el pronóstico de valores futuros o con la finalidad de interpretar eventos ocurridos.

La mayoría de las técnicas matemáticas tradicionales surgidas para el análisis de series temporales se desarrollaron en el siglo XX; un ejemplo de ello es el análisis de regresión. Con el desarrollo de técnicas de pronóstico más complejas junto al advenimiento de máquinas calculadoras y más tarde de computadoras potentes, los pronósticos numéricos recibieron mayor atención. El surgimiento de la minería de datos, y una rama de la misma que se encarga exclusivamente de las series temporales, ha abierto un área de estudio basada en nuevos enfoques y amplias perspectivas de aplicación.

Los métodos utilizados en la minería de datos para series temporales son capaces de caracterizar satisfactoriamente series con características complejas. Estos métodos cubren las limitaciones de las técnicas tradicionales utilizadas en el análisis de series temporales ya que adaptan los conceptos de la minería de datos, para tratar este tipo de series como una clase especial de datos.

En el marco de la colaboración multidisciplinaria existente entre el Instituto de Meteorología de Santa Clara y la Facultad de Matemática Física y Computación de la Universidad Central de Las Villas surge la tarea de modelar el problema del pronóstico de precipitaciones usando técnicas novedosas de minería de datos para series temporales. Al ser el pronóstico de precipitaciones un problema complejo, aún no cuenta con una solución satisfactoria mediante los modelos de pronóstico empleados actualmente en la provincia de Villa Clara. Esto hace que dicho pronóstico sea una temática abierta al estudio, cuya solución es de particular importancia para el país.

Situación problémica

La Minería de Datos se encarga de descubrir patrones desconocidos a partir de grandes volúmenes de datos almacenados. La dependencia temporal entre los mismos es

importante y no se debe soslayar, de ahí la importancia de darle un tratamiento específico. Este es el objetivo principal que persigue la minería de datos para series temporales.

Weka, una de las plataformas para la minería de datos más populares, cuenta actualmente con un paquete dedicado específicamente a la predicción de series temporales. No obstante, las funcionalidades que brinda este paquete resultan insuficientes para enfrentar otras tareas del aprendizaje automatizado. En la literatura se halla un número considerable de propuestas que presentan métodos para la clasificación de series temporales mediante aprendizaje automatizado. Es por esto que contrasta el hecho de que Weka aún no posea algoritmos con esta finalidad.

En el Instituto de Meteorología de la ciudad de Santa Clara se utilizan las salidas numéricas de los modelos de pronóstico globales en la predicción de precipitaciones. Las mediciones de las variables que brindan como salida dichos modelos presentan un ordenamiento temporal, lo cual hace factible la modelación del pronóstico de precipitaciones mediante series temporales. Esta posibilidad, unida a las perspectivas que ofrece la minería de datos para series temporales actualmente, ha motivado la realización de un estudio en este sentido.

Lo expresado anteriormente ha dado como resultado la formulación de las siguientes:

Preguntas de investigación

- ✓ ¿Cómo diseñar un paquete que facilite la clasificación de series temporales haciendo uso del diseño ya establecido en la plataforma de aprendizaje automatizado Weka?
- ✓ ¿Es posible lograr pronósticos de precipitaciones, comparables a los obtenidos actualmente con métodos numéricos, utilizando técnicas de aprendizaje automatizado?

De esta manera se formula el siguiente:

Objetivo general

- ✓ Diseñar e implementar un paquete para Weka que facilite la tarea de clasificación de series temporales, y aplicar tanto los nuevos métodos implementados como los tradicionales al problema del pronóstico de precipitaciones en la provincia de Villa Clara.

Objetivos específicos

- ✓ Diseñar e implementar un paquete para el trabajo con series temporales en Weka que brinde las siguientes facilidades:
  - Reducción de la numerosidad mediante un filtro supervisado.
  - Cálculo de la distancia entre series temporales mediante una métrica elástica.
  - Clasificación de series temporales mediante el algoritmo del vecino más cercano.
- ✓ Modelar el pronóstico de lluvia posibilitando la aplicación de diversas técnicas de la minería de datos:
  - Modelar el problema como una tarea de clasificación.
  - Modelar el problema como una tarea de regresión.

Después de elaborado el marco teórico se plantea la siguiente:

Hipótesis de investigación

- ✓ Las técnicas de minería de datos para series temporales, aplicadas al pronóstico de precipitaciones, ofrecen resultados comparables a los obtenidos mediante los métodos numéricos utilizados en Instituto de Meteorología de la ciudad de Santa Clara.

Estructura general de la tesis

La tesis se estructura en la presente introducción, tres capítulos, conclusiones y recomendaciones.

El Capítulo 1 se dedica a la realización de un marco teórico referente a series temporales. Señalando especialmente las limitaciones que presentan los métodos tradicionales utilizados en su análisis, e introduciendo desde este punto la minería de datos para series temporales como una vía para superar estas limitaciones.

El Capítulo 2 se centra en explicar el diseño e implementación de un paquete con nuevas herramientas desarrolladas especialmente para el análisis de series temporales en Weka. Se realiza además la validación correspondiente a cada una de las herramientas implementadas.

En el Capítulo 3 se aborda el problema del pronóstico de precipitaciones, específicamente aplicando técnicas de minería de datos a las salidas numéricas de un modelo global. Los resultados obtenidos son comparados con los valores de precipitaciones reales medidos por el Instituto de Meteorología de la provincia de Villa Clara.

1 SOBRE LA MINERÍA DE DATOS PARA SERIES TEMPORALES

1.1 Introducción al capítulo

El presente capítulo aborda primeramente los conceptos básicos sobre las series temporales, sus principales características, así como los elementos fundamentales que se han de tener en cuenta durante su análisis. Posteriormente se brinda una breve reseña de los métodos con que se han tratado tradicionalmente las series temporales, destacando sus limitaciones. Partiendo del análisis de estas limitaciones, se realiza una introducción a la minería de datos para series temporales como una vía para superarlas, abordando su fundamento teórico, principales áreas de investigación y los retos que acompañan su estudio. Finalmente se muestra el amplio campo de aplicación que tiene el análisis de series temporales mediante la descripción de tres ejemplos ilustrativos.

1.2 Series temporales

Una serie de tiempo $Z(t)$ es un conjunto de observaciones secuencialmente realizadas en el tiempo, de modo que le corresponde un valor Z_t a cada instante de tiempo t observado [1].

Interesa especialmente el caso en que los valores de la serie están influidos por factores aleatorios. Haciendo uso del lenguaje matemático, una serie de tiempo puede ser considerada como una colección de variables aleatorias $\{Z_t, t \in T\}$ donde T es un conjunto de índices, normalmente el conjunto de los números naturales. Así, los valores de la serie pueden ser vistos como salidas de un proceso estocástico, esto significa que cada valor Z_t de la serie de tiempo puede ser considerado como una observación de una de las variables aleatorias Z_t que integran el proceso. La serie de tiempo de n observaciones sucesivas $(Z_1(t), Z_2(t), \ldots, Z_n(t))$ puede ser considerada como una muestra de una población de series temporales $(Z_1(t), Z_2(t), \ldots, Z_n(t))$ que podían haber sido generadas por un proceso estocástico.

1.3 Análisis de series temporales

En la vida real la mayoría de los fenómenos, que se estudian secuencialmente ordenados en el tiempo, deben tomar en cuenta la dinámica de los procesos con la finalidad de entenderlos de la mejor manera posible. Una herramienta muy útil para alcanzar dicho objetivo es el análisis de series temporales.

Los datos en una serie de tiempo tienen un orden natural, esto hace que su análisis sea un tanto distinto al de otros problemas que no presentan un orden natural en sus observaciones. El análisis de datos mediante series temporales es además distinto del análisis espacial de datos en el cual las observaciones están relacionadas con localizaciones geográficas (por ejemplo, calcular el precio de una vivienda según sus características y ubicación geográfica). Sin embargo, su uso se ha extendido a ramas de la ciencia tan diversas como son la estadística, el procesamiento de señales, reconocimiento de patrones, econometría, matemática financiera, pronóstico climático, electroencefalografía, ingeniería y comunicaciones.

Por ejemplo, en economía se utilizan estas series en el control de la calidad, para estudiar índices de precios en el mercado, desempleo, producto interno bruto (PIB), índices poblacionales etc. En ciencias naturales se utilizan comúnmente para estudiar el nivel de las aguas de ríos y presas, los parámetros meteorológicos, las medidas de poblaciones naturales, etc. Un estudio económico que muestra la correlación causal entre el consumo eléctrico y la producción económica en Australia se puede consultar en el artículo [2].

El análisis de series temporales puede ser visto como la tarea de encontrar patrones en los datos temporales y predecir sus valores. La detección de patrones incluye el análisis de:

- ✓ **tendencias:** Puede ser visto como cambios sistemáticos no repetitivos (lineales o no lineales) de algún valor sobre el tiempo. Un ejemplo podría ser el valor de una acción cuando continuamente esta sube de precio.
- ✓ **ciclos:** Aquí el comportamiento observado durante el tiempo es cíclico.
- ✓ **períodos:** Los patrones detectados se repiten durante un período de tiempo determinado, ya sea anual, mensual o diario (un ejemplo de ello es cuando los volúmenes de venta aumentan en la temporada navideña).
- ✓ **anomalías:** Para ayudar a encontrar patrones, la técnica de detección de anomalías, elimina mucho de los llamados “falsos positivos”.

El objetivo que tradicionalmente ha primado en el análisis de series temporales es el de describir los datos como cierta función en el tiempo que permita analizar con detalles el

pasado y hacer predicciones futuras. Esto se logra estableciendo modelos probabilísticos hipotéticos que representen a los datos. En consecuencia, se lleva a cabo el proceso de ajuste, que incluye desde la estimación hasta la predicción, para finalmente determinar un modelo satisfactorio. Algunos de los objetivos secundarios de este tipo de modelos son el suavizado (más conocido por *smoothing* en inglés), la interpolación y el modelado de estructuras [3].

Los modelos de series temporales deben considerar la naturaleza del fenómeno que describen y determinar los factores que pueden ser incluidos en cada modelo. Por ejemplo, en muchas series económicas es indispensable considerar los efectos estacionales de la serie. Si esto no se toma en cuenta, los modelos obtenidos no serán los apropiados.

Los métodos utilizados en el análisis de series temporales son típicamente divididos en dos clases: los de dominio de frecuencias [4] y los de dominio de tiempo [5]. El primero incluye el análisis espectral y más recientemente el análisis de ondulaciones; el segundo incluye autocorrelación y correlación cruzada. Además, las técnicas de análisis de series temporales pueden ser divididas según sus métodos en paramétricas y no paramétricas. Los enfoques paramétricos asumen que la estacionalidad fundamental del proceso estocástico tiene cierta estructura la cual puede ser descrita usando un reducido número de parámetros (por ejemplo, usando autorregresión o corrimiento de medias). En estos enfoques, el objetivo es estimar los parámetros del modelo que mejor describen el proceso estocástico. Por el contrario, los enfoques no paramétricos estiman explícitamente la covarianza o el espectro del proceso sin asumir que este tenga alguna estructura en particular. Adicionalmente otras clasificaciones han sido creadas para describir series temporales, algunas de ellas son: series lineales y no lineales, univariadas y multivariadas.

1.4 El modelo clásico para el análisis de series temporales

El modelo autorregresivo integrado de media móvil o ARIMA (acrónimo del inglés A*utoregressive Integrated Moving Average*) [6, 7] es un modelo que utiliza variaciones y regresiones de datos estadísticos con el fin de encontrar patrones para efectuar su predicción. Aunque fue desarrollado a finales de los sesenta del pasado siglo, Box y Jenkins [8] lo sistematizaron en 1976, convirtiéndolo de esta forma en la técnica más utilizada en el análisis de series temporales. El método ARIMA involucra hallar la solución de la ecuación diferencial (1.1).

$$\phi_p(B)\phi_p(B^L)x_t = \delta + \theta_q(B)\,\theta_Q(B^L)a_t \qquad (1.1)$$

En esta ecuación, los operadores son seleccionados con órdenes específicos, y los parámetros calculados a partir de los datos de la serie de tiempo usando métodos de optimización, como el de máxima probabilidad [9] o el de mínimos cuadrados [10].

El método ARIMA está limitado por los requerimientos de estacionalidad e inversibilidad de la serie temporal [7], el sistema generador de dicha serie debe ser también invariante y estable. Además, los residuales (las diferencias entre la serie de tiempo y el modelo ARIMA) deben ser independientes y distribuidos de forma organizada [7]. A pesar de que las técnicas de filtrado pueden ser útiles para convertir las series temporales no estacionarias en estacionarias, no siempre es posible cumplir todos estos requerimientos. Además, la mayoría de ellos involucran cálculos complejos y los resultados que se obtienen no siempre son los mejores.

En el mundo real muchas de las series temporales, como por ejemplo las correspondientes a los precios del mercado y al análisis climatológico, no cumplen las condiciones de normalidad residual, estacionalidad y normalidad general de la serie. Un severo inconveniente del método ARIMA es su incapacidad para lidiar con este tipo de complejidades.

Por tanto, solamente con un modelo adecuado, con una correcta identificación de sus parámetros y el supuesto de que la relación entre dichos parámetros es constante en el tiempo, los valores futuros de la serie de tiempo podrán ser pronosticados con un razonable rango de confianza mediante el modelo ARIMA. De no ser así, el modelo resultará inadecuado y los resultados no se corresponderán con la realidad objetiva del fenómeno que se pretende representar.

1.5 El modelo basado en minería de datos para el análisis de series temporales

La minería de datos para series temporales es una contribución importante a los campos de estudio de la minería de datos y de las series temporales. Los métodos utilizados en la minería de datos para series temporales son capaces de caracterizar satisfactoriamente series periódicas, no periódicas, complejas y caóticas. Estos métodos cubren las limitaciones de las técnicas tradicionales utilizadas en el análisis de series temporales, ya que adaptan los conceptos de la minería de datos para tratar este tipo de series como una clase especial de datos. Su campo de estudio utiliza lo mejor de las siguientes áreas

de investigación: análisis estadístico de series temporales, minería de datos, procesado adaptativo de señales, análisis ondulatorio, algoritmos genéticos, sistemas dinámicos y caos.

1.5.1 Minería de datos

Según [11], se puede definir que:

"La minería de datos es el proceso de descubrir nuevas correlaciones significativas, modelos y tendencias, filtrando las grandes cantidades de datos guardadas en repositorios, a través del uso de tecnologías de reconocimiento de modelos así como de técnicas estadísticas y matemáticas".

El objetivo de este proceso es revelar patrones desconocidos a partir de los datos. Su singularidad radica en los tipos de problemas que es capaz de resolver (aquellos con enormes conjuntos de datos y relaciones muy complejas entre ellos). Las metodologías de la minería de datos son derivadas del aprendizaje automático, la inteligencia artificial y la estadística entre otras. El término "descubrir" indica que la información buscada no se puede sacar simplemente con consultas complejas hechas a bases de datos, dada una sospecha de una interrelación en los datos. Tampoco se refiere al uso de pruebas de hipótesis estadísticas usando técnicas estándares de la estadística.

Muchos vendedores de software comercializan su software analítico como aplicaciones que proporcionan soluciones a problemas difíciles sin la necesidad de vigilancia o interacción humana. Algunas definiciones prematuras de la minería de datos siguieron este enfoque de la automatización [12].

En ocasiones el descubrimiento de conocimiento en las bases de datos o KDD (acrónimo del inglés *Knowledge Data Discovery*) se trata como sinónimo de minería de datos. Alternativamente, otros ven la minería de datos como simplemente un paso esencial en el proceso de KDD. Por ejemplo en [13] utilizan el término minería de datos para referirse en general al proceso de descubrimiento de conocimiento a partir de grandes bases de datos, almacenes o repositorios de información.

Es conveniente categorizar la minería de datos en tipos de tareas, correspondiendo a los diferentes objetivos de la persona que va a analizar los datos y los tipos de problemas a que se va a enfrentar. Dada la naturaleza de dichos problemas, los podemos agrupar en distintas tareas tales como: clasificación, agrupamiento, asociación, predicción y regresión [13].

1.5.2 Minería de datos para series temporales

El uso del término de minería de datos implica una analogía con la minería de oro; así como esta última consiste en la búsqueda de pepitas de oro, la minería de datos consiste en la búsqueda de "pepitas de información". Tanto como el oro está oculto bajo tierra o bajo el agua, la información esta oculta dentro los grandes volúmenes de datos. Para un minero inexperto, el oro es simplemente oro, pero para uno veterano, el tamaño de la pepita de oro encontrada marca una diferencia significativa en cómo la excavación será realizada posteriormente. Buscadores individuales usan principalmente métodos manuales cuando buscan pepitas de oro que tienen onzas de peso [14], mientras que las industrias mineras del oro encuentran aceptable la exploración en lugares donde este solo existe a niveles moleculares [14]. Por otro lado, si un explorador anda en busca de plata o petróleo, los procesos de minería serán diferentes.

Esto apunta a la necesidad de tener bien definidas las "pepitas de información" que queremos hallar. La minería de datos para series temporales requiere tener claramente definidos cuáles serán los eventos que vamos a "minar".

La segunda analogía tiene que ver con el modo en el que los exploradores aprenden dónde buscar el oro. Ellos obtienen información geológica según la presencia de metales como el cuarzo o el hierro [14] en el lugar donde suponen la existencia de oro. Este estudio debe ser correcto para no cavar en vano. De manera similar es necesario definir las formaciones que apuntan a eventos significativos. En el contexto de la minería de datos para series temporales estas (probablemente ocultas) formaciones son llamadas patrones temporales. Así como los buscadores de oro entienden que las pistas no tienen necesariamente que ser perfectas, los buscadores de información entienden que las pistas solo necesitan contribuir en alguna medida a propiciar la efectividad de la predicción.

Estas dos analogías nos permiten identificar dos conceptos claves asociados a la minería de datos para series temporales. El primero de todos es el concepto de evento, el cual es una ocurrencia importante. El segundo concepto es el de patrón temporal, una estructura potencialmente oculta en las series temporales. Un patrón temporal es necesario para ayudar a predecir los eventos.

La minería de datos para series temporales constituye una contribución importante al análisis de las series temporales y a la minería de datos. Los métodos basados en este modelo son capaces de predecir y caracterizar con éxito series temporales con características complejas: no periódicas, irregulares y caóticas; superando de esta forma las limitaciones que presentan las técnicas tradicionales. Este tipo de métodos son aplicables a series que parecen estocásticas, y que sin embargo en muchas ocasiones (no

necesariamente de forma periódica) contienen patrones ocultos que caracterizan a un evento determinado.

Se supone comúnmente que en las series temporales modeladas con ARIMA, los cambios en el pasado serán aplicados a la predicción del futuro. Por lo que se asume que estos modelos no necesitarán variar a través del tiempo, que son lineales (y por lo tanto que pueden ser definidos por ecuaciones diferenciales) [15]. Desafortunadamente, el sistema generador de una serie de tiempo no tiene por qué ser necesariamente linear o estacionario.

En contraste con lo anterior, los métodos basados en minería de datos para series temporales son capaces de manipular series temporales no lineales y no estacionarias. Por lo que resultan más útiles para predecir eventos imprevistos en la serie (como por ejemplo el alza repentina del precio de algún producto en el mercado o la rotura de alguna clase de motor en una fábrica etc.).

La naturaleza de las series temporales hace que su tratamiento se diferencie de los métodos tradicionales de minería de datos. Entre estas características distintivas se encuentran: alta numerosidad, gran número de dimensiones y una constante actualización de sus datos al transcurrir el tiempo.

Considerando su naturaleza continua, es imprescindible considerar una serie de tiempo como un todo en lugar tratarla como un conjunto de campos numéricos individuales. A diferencia de otros tipos de datos donde el concepto de similitud se resuelve de forma exacta (en los atributos nominales por ejemplo), para las series temporales el cálculo de la similaridad se satisface de forma aproximada ya que es prácticamente imposible encontrar dos series exactamente iguales.

Los principales conceptos de la minería de datos para series temporales están soportados por importantes campos de investigación, entre ellos la minería de datos [16, 17], el análisis estadístico de series temporales, el procesado adaptativo de señales, el análisis ondulatorio, los algoritmos genéticos, el análisis del caos, y los sistemas dinámicos [18, 19] entre otros.

1.6 Tareas de la minería de datos para series temporales

En los últimos años se han llevado a cabo numerosas investigaciones relacionadas con la minería de datos para series temporales, por ejemplo: el encontrar similitudes entre series temporales [1, 20], la búsqueda de subsecuencias [21], la reducción de su dimensionalidad [22, 23] y la segmentación [24]. Diferentes “tareas de minería de datos

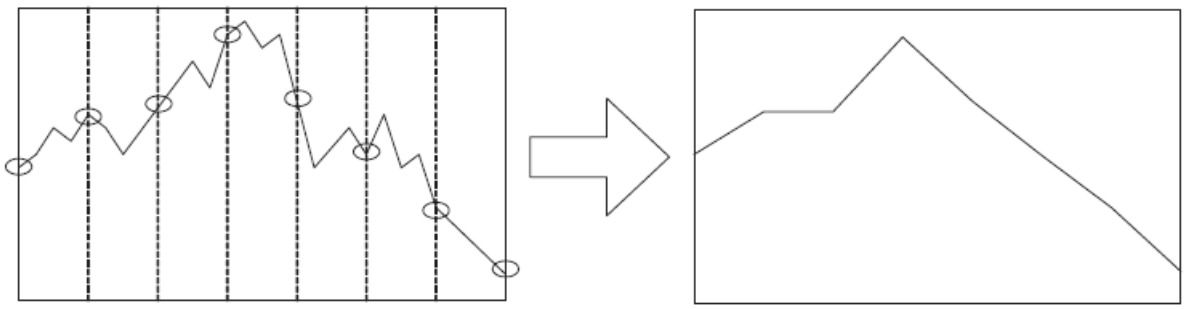
para series temporales" pueden encontrarse en la literatura, varios autores [1], para favorecer su estudio, las clasifican en los siguientes campos:

- ✓ representación e indexado
- ✓ clasificación
- ✓ medidas de similitud
- ✓ emparejamiento de subsecuencias
- ✓ segmentación
- ✓ visualización
- ✓ descubrimiento de patrones y conglomerados

1.6.1 Representación e indexado

Uno de los principales retos de la representación de series temporales es la reducción de su dimensión (por ejemplo, el número de puntos de datos) de la serie original. El método más simple para ello es el muestreo [25]. En este método, una tasa de m/n es usada, donde m es la longitud de la serie de tiempo *P* y n es la dimensión después de efectuada la reducción, Figura 1.1. El método de muestreo tiene la inconveniencia de que distorsiona la forma de la serie temporal obtenida si la tasa de muestreo es demasiado pequeña.

Figura 1.1 Reducción de la dimensionalidad de una serie de tiempo mediante muestreo. La serie de tiempo de la izquierda es muestreada regularmente (denotado por líneas punteadas) y desplegada a la derecha con distorsión alargada.



Un método mejorado consiste en usar el valor medio de cada segmento para representar el correspondiente conjunto de puntos de datos [26]. Otra variante, utilizada en la reducción de la dimensionalidad, consiste en aproximar la serie de tiempo usando líneas rectas [27].

Diversos métodos han sido propuestos para representar las series temporales. Algunos, como los anteriormente analizados, plantean su representación en el dominio del tiempo directamente. La representación de las series temporales en el dominio de transformación constituye otra importante familia de métodos [22, 28, 29].

1.6.2 Clasificación

La clasificación es quizás la técnica más popular de la minería de datos. En el dominio de las series temporales, se debe considerar un tratamiento especial atendiendo a la naturaleza de los datos que se representan.

La clasificación asocia datos entre grupos predefinidos o clases. La mayoría de los algoritmos de clasificación asumen algún conocimiento de los datos o realizan fases de entrenamiento para estas clasificaciones. El problema de la clasificación de series temporales puede ser definido de la siguiente forma:

Dada una base de casos $D = \{t_1, t_2, \dots, t_n\}$ *constituida por series temporales, y un conjunto de clases* $C = \{C_1, C_2, \dots, C_m\}$*, el problema de clasificar dichas series es el de definir una función* $f: D \rightarrow C$ *donde cada* t_i *es asignada a una clase. Y una clase* C_j *contiene precisamente a las series asignadas en ella, esto es* $C_j = \{t_i \mid f(t_i) = C_j\ ,\ 1 \leq i \leq n, t_i \in D\}$.

En [32] se propone clasificar los datos de la serie de tiempo basándose en sus propiedades locales o en sus patrones. En [33] se desarrolló un método de representación usando descomposición ondulatoria que puede seleccionar automáticamente los parámetros para la clasificación. Además, en [34] se proponen clasificadores de series temporales semi-supervisados para los que solo son necesarios un grupo reducido de ejemplos etiquetados de aprendizaje.

Los investigadores se han enfocado últimamente en desarrollar clasificadores a la medida. Por ejemplo, en [35] se presenta una investigación sobre la clasificación de señales basándose en el modelado de un sistema dinámico que captura los datos para la serie usando modelos de texturas Gaussianos. En [36] se estudia la adecuación de la medida de distancia DTW y de los árboles de decisión para la clasificación, mientras que en [37] se estudia la combinación de la reducción de la numerosidad usando DTW y los clasificadores basados en el vecino más cercano.

En el caso de esta última técnica, el algoritmo de los k vecinos más cercanos (kNN[1]), a pesar de su simplicidad, es uno de los que mejores resultados ha ofrecido para la clasificación de series temporales. kNN está basado en el uso de métricas, y asume que los datos de entrenamiento son en sí el modelo de los datos. Cuando se tiene que hacer una clasificación dado un nuevo conjunto de datos, se calcula su distancia con cada uno de los elementos en el modelo. La serie de tiempo es asignada a la clase mayoritaria que contiene los k elementos más cercanos a ella. Entonces, dado que cada serie de tiempo debe ser clasificada y considerando que existen q elementos en el modelo, esto significa que tiene complejidad de orden $O(q)$. Teniendo n elementos a clasificar, la complejidad llega a ser del orden $O(nq)$, pero como el tamaño del modelo es constante, se puede ver en forma generalizada como una complejidad de $O(n)$.

1.6.3 Medidas de similitud

Las medidas de similitud tienen una gran importancia para las distintas tareas del análisis de series temporales. No resulta en absoluto trivial definir funciones de similitud dada la naturaleza numérica y continua de las series temporales. Existen dos enfoques principales para el cálculo de la similitud: considerar la serie de tiempo en toda su longitud, y la comparación de subsecuencias. Una de las distancias más usadas es la distancia euclidiana [38], que se emplea fundamentalmente en las series temporales transformadas. Aunque varios estudios revelan que no siempre es la distancia indicada para dominios más específicos [39].

Una de las medidas de similitud más populares usada actualmente se conoce con el nombre de distorsión dinámica del tiempo o DTW (acrónimo del inglés *Dynamic Time Warping*) [40]. Con esta técnica se consigue un alineamiento optimal entre dos series temporales emparejándolas de forma no lineal mediante contracciones y dilataciones de las series en el eje temporal. Por consiguiente, este emparejamiento permite encontrar regiones equivalentes entre las series o hallar su similitud.

DTW ha encontrado aplicación en varias disciplinas como son: minería de datos, reconocimiento de gestos, robótica, en procesos fabriles o en medicina [28]. En el reconocimiento del lenguaje oral, donde tuvo su primera aplicación, esta medida de distancia resulta útil para determinar si dos ondas sonoras representan la misma frase en

[1] Se representa como kNN al algoritmo de los *k* vecinos más cercanos cuando este usa como métrica la distancia euclidiana; cuando k=1 suele representarse como 1NN.

una conversación interpersonal cualquiera. Esto se debe a que la duración del sonido de cada letra puede variar, pero la onda sonora en general debe tener la misma forma para la misma frase. En minería de datos para series temporales DTW es usada comúnmente como una función de distancia.

Para alinear dos series temporales *P* y *Q* usando DTW, primeramente se construye una matriz M_{nxm}. El i-ésimo elemento de la matriz m_{ij}, contiene la distancia $d(q_i, p_j)$ entre dos puntos q_i y p_j, y la distancia euclidiana es la típicamente usada:

$$d(q_i, p_j) = (q_i - p_j)^2 \tag{1.2}$$

Esto corresponde con el alineamiento entre los puntos q_i y p_j de las series. Un camino distorsionado *W* es un conjunto continuo de elementos de la matriz que definen una correspondencia entre *P* y *Q.* El k*-ésimo* elemento se define como:

$$w_k = (i_k, j_k) \tag{1.3}$$

$$W = w_1, w_2, \dots, w_k, \dots, w_K$$

Donde se cumple:

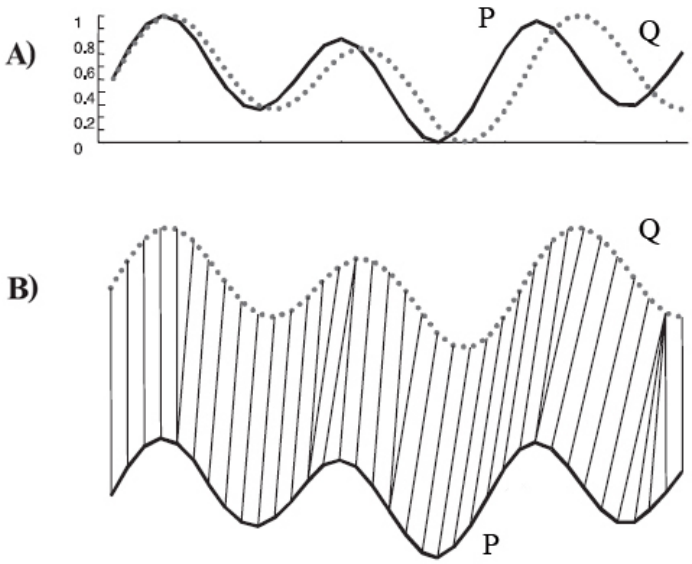
$$max(m,n) \leq K < m + n - 1 \tag{1.4}$$

El camino distorsionado esta comúnmente sujeto a varias restricciones, como por ejemplo: las condiciones de frontera, continuidad y monotonía. Las restricciones de frontera son $w_1 = (1,1)$ y $w_k = (m,n)$. Esto hace que el camino distorsionado comience y termine diagonalmente.

Otra restricción es la de la continuidad. Dado $w_k = (a,b)$, entonces $w_{k-1} = (a', b')$, donde $a - a' \leq 1$ y $b - b' \leq 1$. Esto restringe los pasos posibles en el camino a las celdas adyacentes, incluyendo la celda diagonal adyacente. Además, las restricciones $a - a' \leq 1$ y $b - b' \leq 1$ fuerzan a los puntos en *W* a ser monótonamente espaciado en el tiempo.

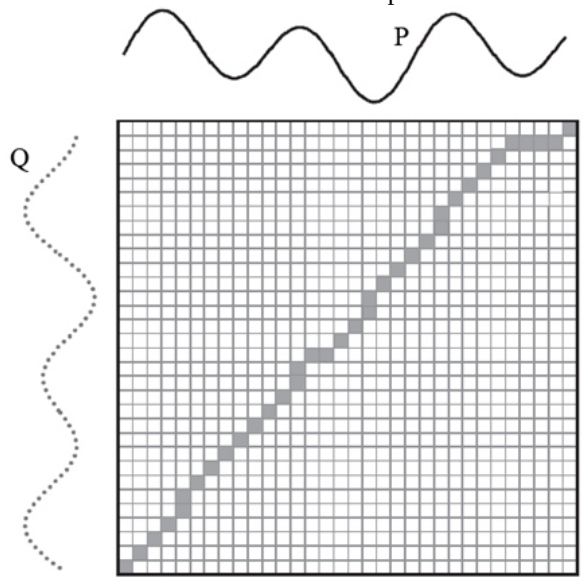
La Figura 1.2 muestra como dos series *P* y *Q* son emparejadas entre sí a lo largo del tiempo. Cada punto de la serie *Q* es conectado con su correspondiente punto similar en la serie *P* mediante una línea recta que los une. Si ambas series en la figura fuesen idénticas, todas las líneas entre ellas serían verticales pues no se necesitaría de un emparejamiento diferente para alinearlas entre sí. La distancia DTW es una medida de la diferencia entre dos series temporales después de que ambas hayan sido alineadas de forma óptima. Dicha medida se corresponde con la suma de las distancias entre cada par de puntos conectados.

Figura 1.2 A) Dos series temporales *P* y *Q* similares pero desfasadas. **B)** El emparejamiento entre los puntos de cada serie usando DTW permite detectar desfasajes en el tiempo.



Existen un número bastante grande de caminos que satisfacen las condiciones antes mencionadas. Sin embargo, solo nos interesa el camino que minimice el costo de alineamiento, Figura 1.3.

Figura 1.3 Matriz ***M*** donde se forma el camino ***W*** mínimo para los alineamientos entre las series *P* y *Q*.



Este camino puede ser eficientemente hallado usando programación dinámica. Para ello se evalúa la ecuación de recurrencia (1.5), que define la distancia acumulada $\gamma(i,j)$ como

la distancia $d(i,j)$ encontrada en la celda actual y el mínimo de las distancias acumuladas de los elementos adyacentes:

$$\gamma(i,j) = d(q_i, p_j) + min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\} \quad (1.5)$$

Un camino de alineamiento W, tal que la distancia entre dichos elementos adyacentes sea mínima, puede calcularse mediante la ecuación (1.6).

$$DTW(Q,P) = min\left\{\sqrt{\sum_{k=1}^{k} d(w_k)}\right\} \quad (1.6)$$

Donde $d(w_k)$ $d(w_k)$ puede definirse como muestra la ecuación (1.7). Detalles sobre este tratamiento pueden verse en [41].

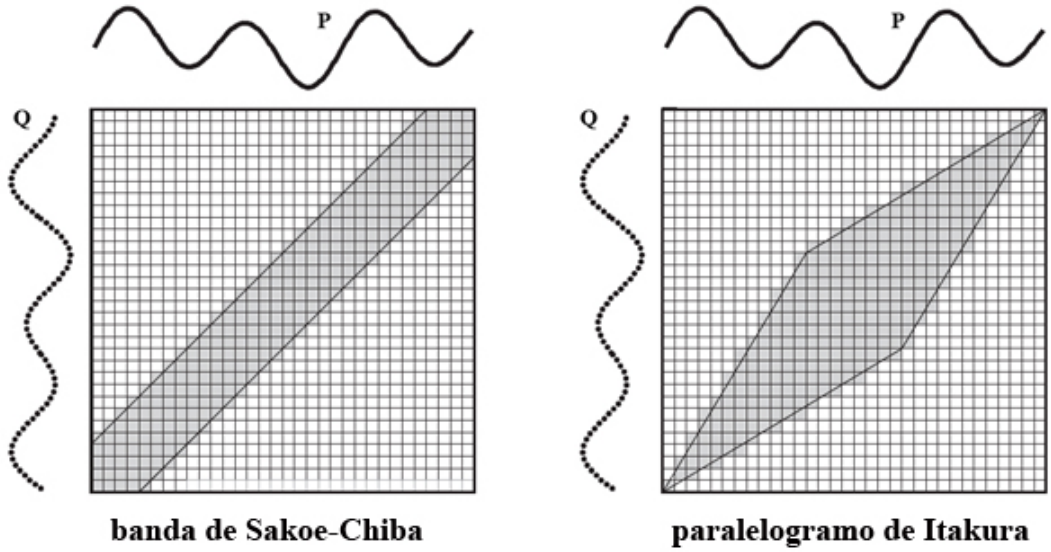
$$d(w_k) = d(q_{ik} - p_{ik})^2 \quad (1.7)$$

El cálculo de DWT tiene una complejidad temporal y espacial de orden cuadrático $O(n^2)$ lo cual hace que su implementación computacional sea costosa cuando los puntos de datos de las series superan los miles. Diferentes métodos han sido propuestos para acelerar la velocidad del proceso del cálculo de las distancias DTW, estos pueden ser divididos en las siguientes categorías:

- ✓ **restricciones globales:** Limitar el número de celdas que son evaluadas en la matriz de costos (banda Sakoe-Chiba, paralelogramo de Itakura).
- ✓ **abstracción de los datos:** Ejecutar DTW en una reducida representación de los datos (reducción de la numerosidad).
- ✓ **indexado:** Usar funciones de acotación para reducir el número de veces que DTW deberá ejecutarse durante la clasificación o el agrupamiento (por ejemplo, el uso del algoritmo LB_Keogh para la acotación inferior de DTW).

El establecimiento de restricciones globales que controlan el subconjunto de la matriz que el camino de alineamiento es capaz de visitar, Figura 1.4, es uno de los métodos más usados en el mejoramiento de la eficiencia de DTW [42, 43].

Figura 1.4 Dos de las restricciones globales más usadas. El subconjunto de la matriz que el camino de alineamiento es capaz de visitar es también conocido como ventana *warping*.



De esta forma, se restringen los índices del camino $w_k = (i,j)_k$ tal que $j-r \leq i \leq j+r$, donde r es el término que regula la elasticidad permitida para un punto dado de la serie (normalmente r es pasada como parámetro de la función que calcula DTW). En el caso de la banda Sakoe-Chiba, r es independiente de i; para el paralelogramo de Itakura r es una función de i. Evaluaciones empíricas en numerosos conjuntos de datos han demostrado que los mejores resultados se obtienen cuando el valor de r no supera el 10% de la longitud de las series.

Otro método de probada eficacia para el mejoramiento de la eficiencia del cálculo de DTW es el uso de la cota inferior LB_Keogh, con lo cual se evitan numerosos cálculos innecesarios de DTW. Para ello, se definen dos nuevas series en base a r:

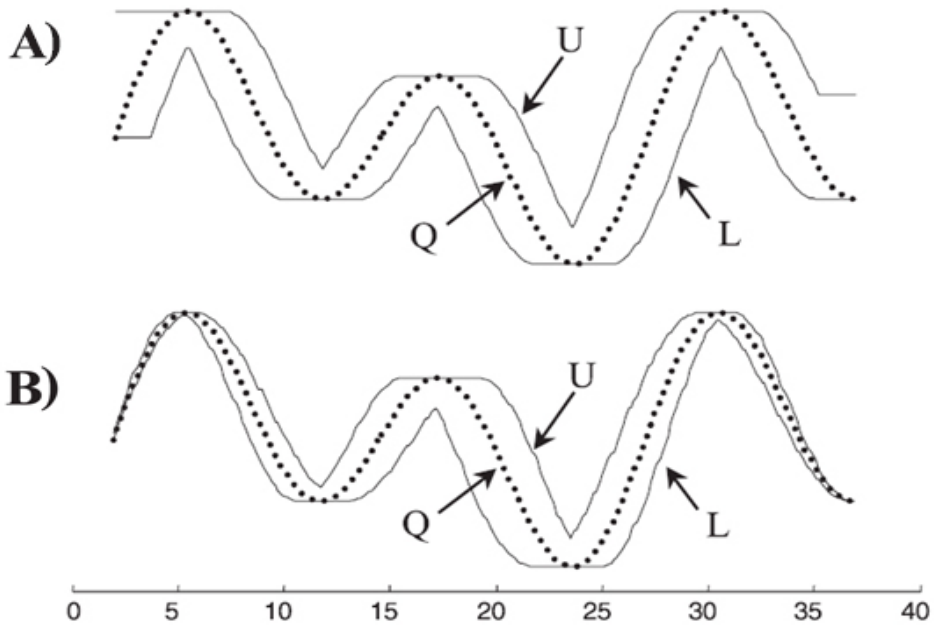
$$U_i = max(q_{i-r}, q_{i+r}) \qquad (1.8)$$

$$L_i = min(q_{i-r}, q_{i+r})$$

U y *L* representan la serie superior (*Upper*) e inferior (*Lower*) respectivamente de una serie *Q* dada. Como se puede apreciar en la Figura 1.5, ambas series conforman una banda que cubre totalmente la serie *Q*. Notar que, aunque la banda de Sakoe-Chiba tiene un ancho constante en la matriz, la banda correspondiente que envuelve la serie generalmente no tiene un espesor uniforme. En particular, dicha envolvente se hace más

ancha en los puntos en los que la serie cambia más rápidamente, y se achica en sus mesetas.

Figura 1.5 Una ilustración de las series *U* y *L* creadas para la serie *Q*. **A)** usando la banda de Sakoe-Chiba y **B)** usando el paralelogramo de Itakura.



Una propiedad obvia pero importante de las series *U* y *L* es la (1.9).

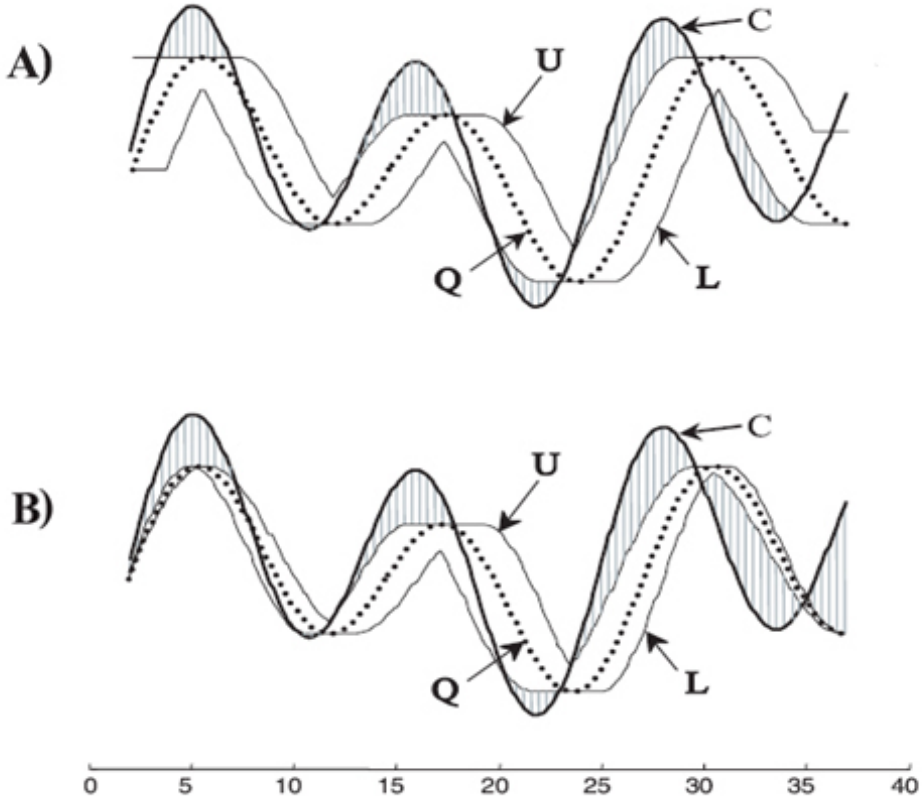
$$\forall_i: \ \mathrm{U}_i \geq q_i \geq \mathrm{L}_i \qquad (1.9)$$

Habiendo definido las series *U* y *L*, se define la siguiente medida de acotación inferior para DTW:

$$LB\_Keogh(P,Q) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - \mathrm{U}_i)^2 & si \ \ c_i > \mathrm{U}_i \\ (c_i - \mathrm{L}_i)^2 & si \ c_i < \mathrm{L}_i \\ 0 & en \ otro \ caso \end{cases}} \qquad (1.10)$$

Esta función puede ser visualizada como la distancia euclidiana entre cualesquiera de los puntos de la serie candidata y la envolvente más cercana a dicha serie, Figura 1.6.

Figura 1.6 LB_Keogh calcula el cuadrado de la suma de la distancia euclidiana que hay entre cualquier parte fuera de la envolvente de la serie candidata *C*, al borde ortogonal más cercano de la envolvente definida por *L* y *U*. Este valor es devuelto como una cota inferior de DTW. **A)** Banda de Sakoe-Chiba, **B)** Paralelogramo de Itakura



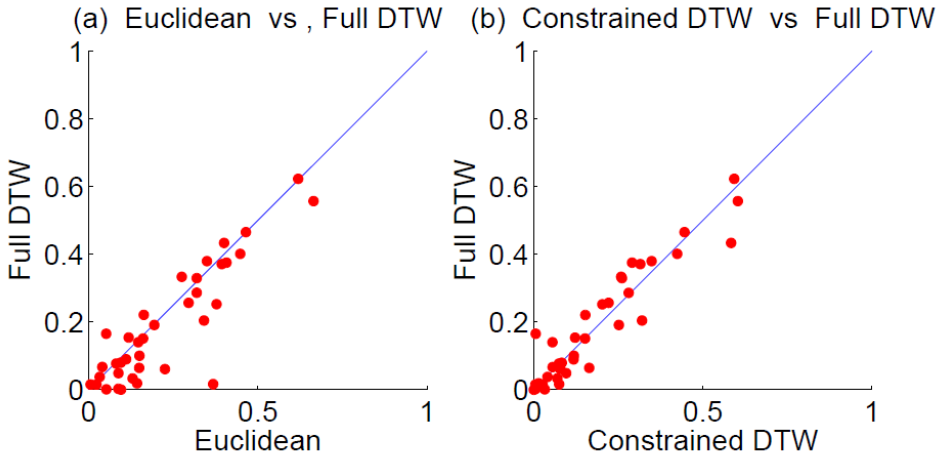
De aquí se puede probar (1.11) [44].

$$\forall_i\colon \ \mathrm{U}_i \geq q_i \geq \mathrm{L}_i, \quad LB\_Keogh(P,Q) \leq DTW(P,Q) \tag{1.11}$$

Por tanto, LB_Keogh establece una cota inferior de DTW, y con ello, se evitan numerosos cálculos innecesarios a lo hora de calcular dicha distancia. Cabe señalar que el costo computacional del cálculo de esta cota inferior es $O(n)$, o sea, un costo lineal mucho menor que el costo $O(n^2)$ que conlleva el cálculo de DTW.

En [45] se lleva a cabo una evaluación general de DTW comparándola otras métricas de distancia. La Figura 1.7 muestra los resultados obtenidos para una colección de 38 conjuntos de datos de series temporales en la cual cada punto rojo representa un conjunto de datos, y los ejes de coordenadas los índices de error de cada uno. La comparación es por pares “A contra B”, un punto rojo bajo la línea diagonal indica que “A” es superior a “B”:

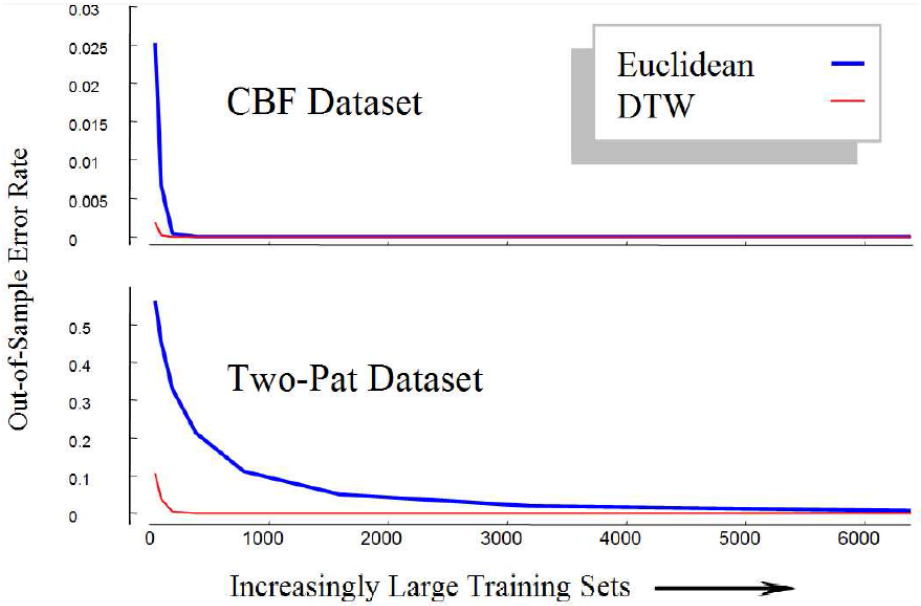
Figura 1.7 Comparación entre las medidas de similaridad **A)** euclidiana contra DTW sin restricciones globales **B)** DTW sin restricciones globales contra DTW con un ancho de ventana igual al 10% del conjunto de datos.



Como se muestra en la figura anterior, la distancia DTW sin restricciones globales es claramente superior a la euclidiana. Además se aprecia que DTW con un ancho de ventana igual al 10% del conjunto de datos (el cual no tiene por qué ser el tamaño óptimo) es casi igual (e incluso ligeramente superior) a DTW sin restricciones globales. Por lo que se valida el uso de restricciones globales en lugar de realizar el cálculo completo de DTW, reduciendo de esta forma el costo computacional empleado en su cómputo.

Por otro lado, se ha comprobado empíricamente que tanto los porcentajes de clasificación correcta como la velocidad de los cálculos amortizados dependen en gran medida del tamaño del conjunto de datos a utilizar. Como una forma de ilustrar la afirmación anterior, en [45] se realizaron experimentos individuales en los conjuntos de datos Two Patterns y CBF (por ser estos los más utilizados en la literatura para comprobar la superioridad de una u otra medida de similitud). Debido a que ambos son conjuntos de datos sintéticos, es posible generar tantas instancias de ellos como sea deseen, por lo que existen versiones con múltiples tamaños. Se midieron los índices de errores de clasificación usando 1NN obtenidos con la distancia euclidiana y con DTW como muestra la Figura 1.8. Como se puede apreciar, la tasa de error relativo de la distancia DTW es significativamente menor a la euclidiana, sobre todo cuando los conjuntos de datos son menos numerosos.

Figura 1.8 Tasa de error relativo para 1NN usando la distancia euclidiana y DTW al incrementar la numerosidad de las series en dos conjuntos de datos tradicionales.



Para CBF, cuando las series temporales en el conjunto de datos superan las 400, no hay diferencias estadísticas significativas entre una y otra métrica de distancia. En el caso de Two Patterns, la distancia euclidiana necesita de un aumento mucho mayor en la numerosidad del conjunto de datos para converger a la precisión de DTW.

Esto pudiera parecer desalentador, teniendo en cuenta que la distancia euclidiana tiene una complejidad temporal de $O(n)$ y que un cálculo simple de DTW tiene una complejidad de $O(nw)$, donde w es al ancho de la ventana usada. No obstante la complejidad amortizada de DTW durante la clasificación es realmente de $O((P * n) + (1 - P) * nw)$, donde P es la fracción de cálculos de DTW podados usando el algoritmo LB_Keogh para la búsqueda de vecinos más cercanos en el algoritmo 1NN.

1.6.4 Emparejamiento de subsecuencias

Dadas una secuencia de entrada y una serie de tiempo de mayor longitud, la tarea en este caso es hallar la subsecuencia en la serie de tiempo que se "empareje" mejor con una secuencia dada. Los primeros trabajos sobre este tema se pueden revisar en [46] y [21]. A partir de estos trabajos, numerosas investigaciones se han llevado a cabo para mejorar el funcionamiento de la búsqueda de subsecuencias. Por ejemplo, los métodos DualMatch [47] y GeneralMatch [48] proponen el uso de ventanas móviles y [49]

desarrolla un algoritmo de ordenamiento de la secuencia para reducir el número de subsecuencias a las cuales se necesita tener acceso durante el emparejamiento.

1.6.5 Segmentación

La segmentación puede ser vista tanto como un paso de preprocesado para numerosas tareas de la minería de datos o como una técnica de análisis de tendencia. También puede ser considerada como un proceso de discretización. En [50] se propone un método simple de discretización. Una ventana de longitud fija es usada para segmentar la serie de tiempo en subsecuencias y de esta forma representarla mediante patrones primitivos. Este proceso depende fundamentalmente de la elección del ancho de la ventana. Existen al menos dos desventajas significativas. Primero, los patrones fundamentales aparecen típicamente con diferentes longitudes a través de toda la serie. Segundo, como un resultado de la segmentación de cualquier serie de tiempo, los patrones más importantes pueden perderse cuando se separan datos en el tiempo.

Por tanto, es preferible usar enfoques dinámicos que identifiquen los puntos de datos que podemos dividir en el tiempo antes de proceder al segmentado de la serie. La tarea de segmentación descrita anteriormente puede ser vista también como un problema de optimización.

1.6.6 Visualización

La visualización es un importante mecanismo para presentar la serie de tiempo procesada, ya que de esta forma se facilita su análisis a los usuarios. Es además una poderosa herramienta que hace más factible las tareas de minería de datos en la serie. Algunos de los productos de software más importantes desarrollados en este sentido son: TimeSearcher [52, 53] y VizTree [31]. Ambos incluyen:

- ✓ visualización de calendarios y conglomerados [54].
- ✓ visualización en espiral [55].

1.6.7 Descubrimiento de patrones y conglomerados

El descubrimiento de patrones, también llamado descubrimiento causal de patrones o detección de anomalías, es la tarea no trivial de descubrir patrones [44] interesantes en

una serie de tiempo. Dada su importancia, se ha convertido en una de las tareas fundamentales de la minería de datos para series temporales y puede ser aplicada a numerosos dominios de investigación [56-58]. Variadas técnicas se han desarrollado, entre ellas cabe señalar: el algoritmo de Gecko [59], las técnicas basadas en distancia [50, 60, 61], el modelo basado en cadenas de Markov para series temporales [62] y el método de agrupamiento neuronal de conglomerados para el reconocimiento autoorganizado [63].

1.7 Principales campos de aplicación y algunos problemas representativos

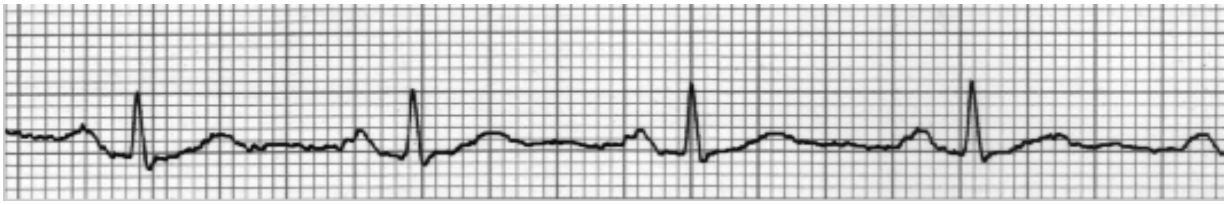
El repositorio de series temporales conocido como UCR (acrónimo del inglés *University of California Riverside*) [64] ha sido creado como un servicio público para la comunidad científica que trabaja la minería de datos y el aprendizaje automatizado. Su objetivo es alentar las investigaciones en el campo de la clasificación de series temporales. En este sitio se ponen a disposición de los investigadores más de 50 conjuntos de datos internacionales de probada fiabilidad, así como información valiosa sobre los mismos (sus creadores, la cantidad de instancias que contienen, una descripción de sus clases, los mejores resultados obtenidos de cada uno de ellos con diversos algoritmos de clasificación y varias medidas de similitud etc.).

En UCR se publican además los más novedosos artículos científicos sobre el tema, así como el código fuente de numerosos algoritmos tradicionales. Durante muchos años se ha invitado a la comunidad científica cuyo campo es la minería de datos a que contribuyan con nuevos conjuntos de datos para el sitio, esto se ha realizado con el objetivo de que la colección existente represente los intereses de grupos cada vez más diversos de investigadores, y no de algunos en particular. Debido a la utilidad que tienen los conjuntos de datos para la validación experimental de las investigaciones, este epígrafe muestra algunos de ellos; siendo ejemplos además, de la diversidad de aplicaciones que tiene actualmente el campo de estudio de la minería de datos para series temporales.

1.7.1 ECG200

La clasificación de enfermedades cardiacas ha recibido una gran atención de la comunidad científica por la gran cantidad de datos disponibles libremente y su potencial aplicación en la medicina, Figura 1.9.

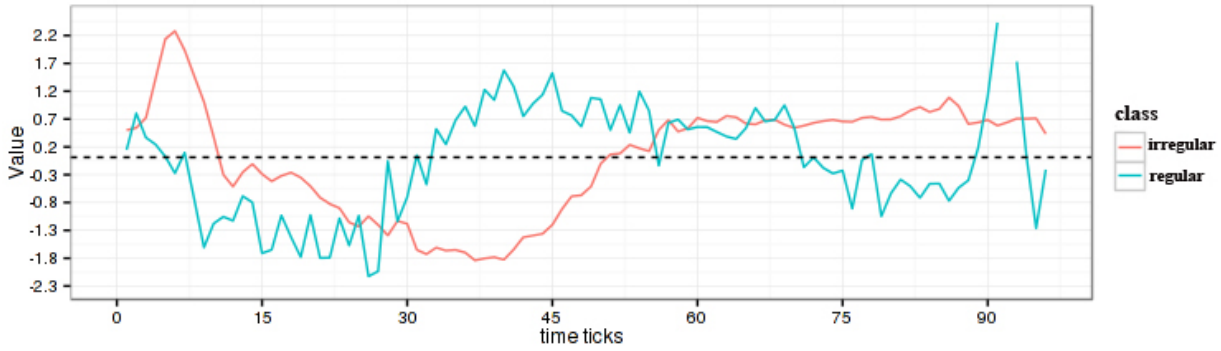
Figura 1.9 Ejemplo de un electrocardiograma.



El Instituto Nacional de Metrología de Alemania ha provisto la compilación de un gran conjunto de datos ECG [2] para su investigación con algoritmos de medición computacionales. La información de ECG fue recogido de voluntarios sanos y otros con algún tipo de enfermedad cardiaca. Cada dato almacenado en ECG es una serie de tiempo registrada por un electrodo durante cada pulsación del corazón. Los datos han sido anotados por cardiólogos y dos clases han sido definidas: regular e *irregular*, para el caso de un comportamiento normal y otro anormal respectivamente. De las 200 instancias del conjunto de datos, 75 fueron identificadas por los especialistas como anormales, y 125 como normales. Todos los datos han sido normalizados y reescalados para que tengan longitud 95 (Recientes resultados sugieren que no se pierde nada con el reescalado [65]). La Figura 1.10 muestra la comparación entre dos series temporales que representan a dos instancias ECG de diferente clase.

[2] El **electrocardiograma** (**ECG**/**EKG**, del alemán *Elektrokardiogramm*) es la representación gráfica de la actividad eléctrica del corazón, que se obtiene con un electrocardiógrafo en forma de cinta continua.

Figura 1.10 Ejemplo de dos clases del conjunto de datos ECG200.

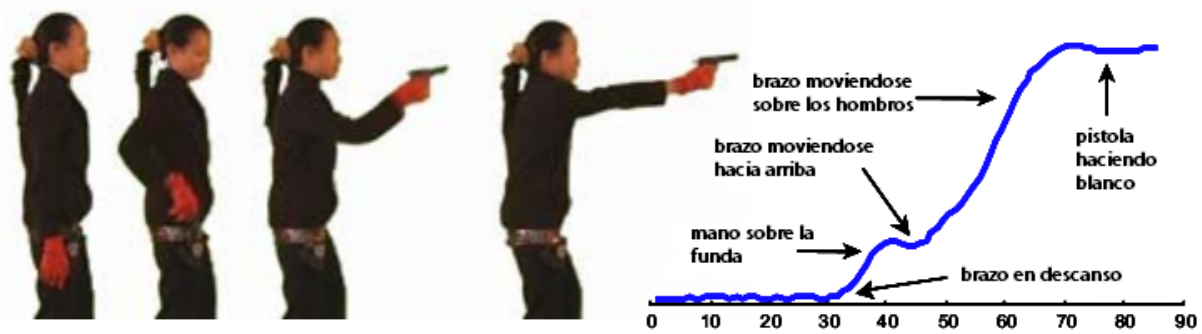


Debido a que los cardiólogos están más interesados en las ocurrencias anormales de los electrocardiogramas, el objetivo de la minería de datos para este problema es conocer si una instancia dada es clasificada en normal o anormal, precentando una mayor importancia aquellas que son clasificadas como anormales.

1.7.2 Gun Point

El conjunto de datos Gun Point pertenece al dominio de los videos de vigilancia. Gun Point consta de dos clases, cada una contiene 200 instancias, 100 para cada clase. Todas las instancias fueron creadas usando un actor del sexo masculino y uno del sexo femenino, durante una única sesión de pruebas en la que los actores fueron grabados en un video (a 30 cuadros por segundo), como se muestra en la Figura 1.11.

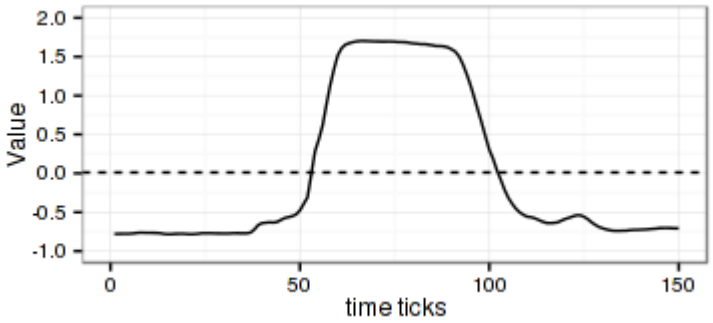
Figura 1.11 Instantáneas de una secuencia de video; el movimiento de la mano derecha es rastreado y convertido en una serie de tiempo.



La grabación fue fácilmente segmentada en 150 puntos de datos que representan una instancia. Las dos clases son:

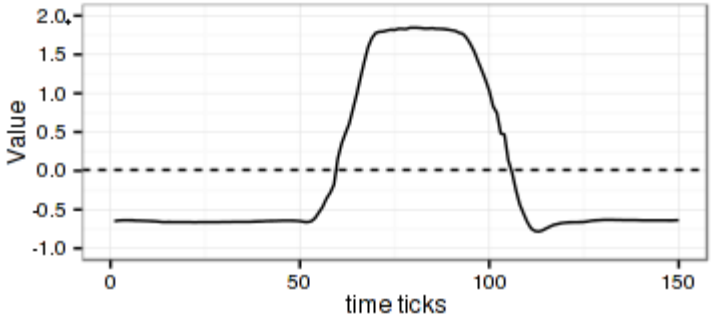
- ✓ **Gun:** Primeramente, los actores sitúan ambas manos a los lados del cuerpo. Luego levantan un arma desde su funda ubicada en su cintura hasta que la pistola se posiciona para hacer blanco (esto ocurre en aproximadamente un segundo). A continuación vuelven a poner el arma en su funda y ambas manos a los lados del cuerpo, Figura 1.12.

Figura 1.12 Ejemplo de una instancia de la clase Gun



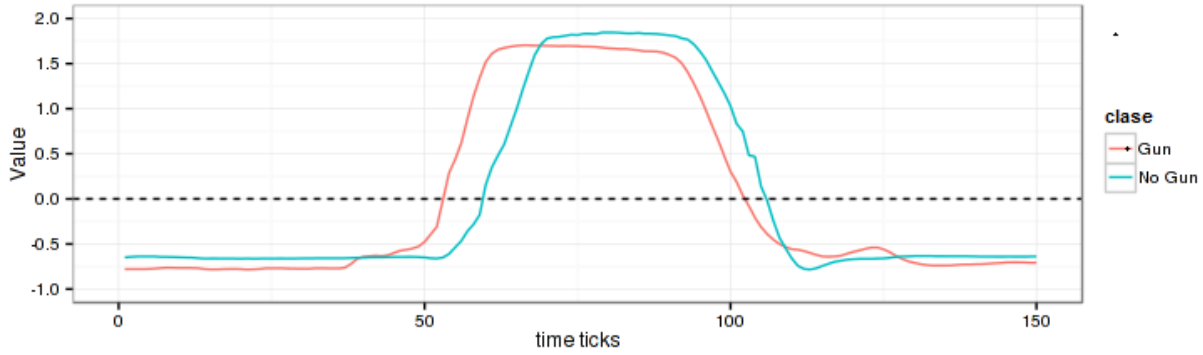
- ✓ **No Gun**: Los actores mantienen su pistola en su funda. Apuntan con el dedo índice a un objetivo aproximadamente en un segundo, para más tarde volver a situar sus manos a ambos lados del cuerpo, Figura 1.13.

Figura 1.13 Ejemplo de una instancia de la clase No Gun.



El centroide de la mano derecha del actor es capturado por la cámara en los ejes X y Y; los cuales parecen estar muy correlacionados. En este experimento solo se considera el eje X por simplicidad. Para más detalles sobre el experimento consultar [65].

Figura 1.14 Ejemplo de dos clases del conjunto de datos Gun Point.

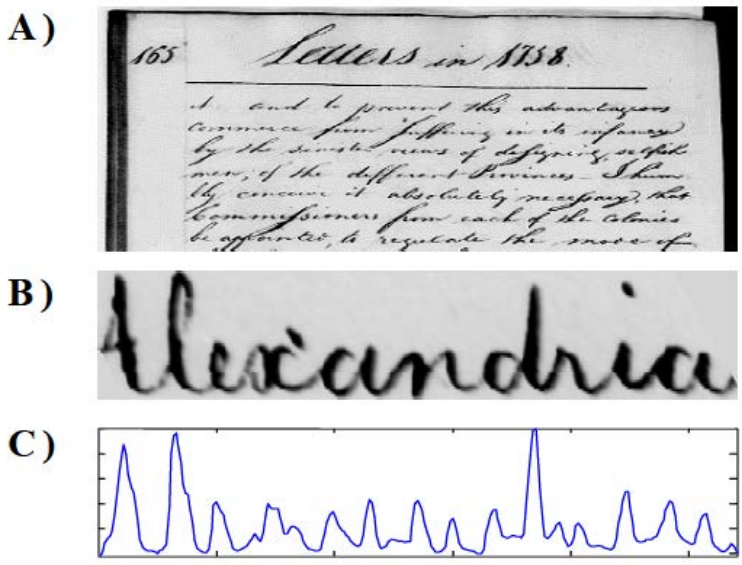


En este problema el objetivo consiste en conocer si el actor apunta o no con un arma mediante la caracterización de cada uno de sus movimientos. Los eventos considerados relevantes son tomados mediante las observaciones de la cámara y corresponden al movimiento de la mano derecha del actor. Aunque las clases sean muy similares entre sí, Figura 1.12 y Figura 1.13, es posible clasificar visualmente ambas clases con gran precisión después de notar que el actor debe alzar su mano sobre la funda de su pistola para sacarla. Esta acción genera una sutil distinción entre ambas clases la cual se hace visible fácilmente en la Figura 1.14.

1.7.3 Fifty Words

El problema de transcribir e indexar archivos históricos existentes es aún un reto. Incluso para figuras históricas de la talla de Isaac Newton existen una gran cantidad de trabajos escritos a mano que todavía no han sido publicados (los trabajos de Newton exceden el millón de palabras). Para otras muchas personalidades, están recogidos actualmente muchos trabajos escritos a mano, colecciones que tienen un valor incalculable para biógrafos e investigadores, y que todavía no han sido descifrados y traducidos enteramente debido a la complejidad de este problema. Sorprendentemente, es posible transformar texto escrito a mano en series temporales, Figura 1.15.

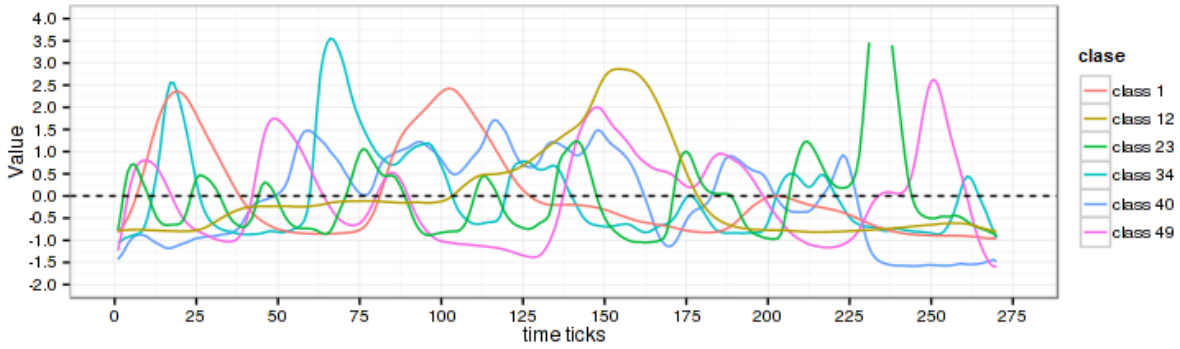
Figura 1.15 A) Ejemplo de un texto escrito por George Washington. **B)** La palabra "*Alexandria*" del texto A) luego de haber sido procesada para eliminar su inclinación. **C)** La palabra de B) convertida en una serie de tiempo.



Por ejemplo, consideremos el problema de traducir textos bíblicos a dos lenguajes diferentes [65]: inglés y español. Para ello, el texto bíblico es convertido por entero en cadenas de bits de acuerdo a las ocurrencias de una palabra seleccionada en el texto. Por ejemplo, un apartado de la Biblia en español que contiene la palabra "Dios" en la frase "En el comienzo Dios creo el cielo y la tierra" será representada como "0001000000". Esta cadena de bits es convertida entonces en una serie de tiempo registrando el número de ocurrencias de la palabra dentro de una ventana móvil predefinida para todo el texto. Intuitivamente, por cada aparición de la palabra en idioma inglés, debe estar presente su correspondiente en español. Sin embargo, pueden existir algunas discrepancias en el número de palabras existentes en el texto completo, así como en la posición de la palabra dentro de cada oración para ambos lenguajes. Estas irregularidades pueden ser analizadas en detalle usando técnicas de minería de datos.

El conjunto de datos conocido como Fifty Words, fue creado por Rath y Manmatha para el emparejamiento de imágenes [66]. Contiene 2381 imágenes de palabras, de 10 páginas escritas. Se han tomado imágenes de 50 palabras comunes en idioma ingles como "the", "and", etc. obteniéndose 905 instancias en total. Cada imagen de palabra es representada por una serie de tiempo cuatridimensional la cual describe las características de la imagen, Figura 1.15. Por ejemplo en la Figura 1.16 se muestra el perfil de la palabra "Alexandria". Por simplicidad, Fifty Words considera solo la primera dimensión de cada imagen, la cual tiene una longitud promedio de 270.

Figura 1.16 Ejemplo de seis clases del conjunto de datos Fifty Words.



Luego, el problema para este conjunto de datos se centra en diferenciar la palabra “the” del resto. En total, existen 109 imágenes de la palabra “the” y 796 imágenes de las otras palabras, para un total de 905 imágenes en todo el conjunto de datos.

1.8 Conclusiones del capítulo

Las series temporales permiten describir de forma natural gran variedad de fenómenos que transcurren a lo largo del tiempo. Es por ello que su uso se ha extendido a numerosas áreas del conocimiento, especialmente aquellas que requieren predecir el comportamiento de determinadas variables de interés en un momento dado. Aplicar los modelos estadísticos tradicionales en el análisis de series temporales precisa del cumplimiento de ciertos requerimientos, lo cual es una limitación no desdeñable en muchos casos. Los modelos que analizan las series temporales usando técnicas de minería de datos son capaces de resolver problemas con estas limitaciones. Dada la gran cantidad de aplicaciones prácticas que han surgido, actualmente existe un incremento de los estudios que se realizan en el campo de la minería de datos para series temporales; los cuales están divididos en las siguientes categorías: representación e indexado, clasificación, medidas de similitud, emparejamiento de subsecuencias, segmentación, visualización, descubrimiento de patrones y descubrimiento de conglomerados.

2 IMPLEMENTACIÓN DE UN PAQUETE PARA LA CLASIFICACIÓN DE SERIES TEMPORALES EN WEKA

2.1 Introducción al capítulo

En este capítulo se ofrece una breve introducción al ambiente de aprendizaje automatizado Weka[3] y a lo que este brinda para el trabajo con series temporales; destacando sus ventajas y desventajas, así como la necesidad de extenderlo incorporándole nuevas funcionalidades. Se describen el diseño y la implementación de un paquete, con nuevas herramientas desarrolladas especialmente para el análisis de series temporales en Weka. Se explica por qué es necesario su uso y las características principales de los recursos computacionales utilizados en su desarrollo; haciéndose énfasis en las ventajas que ofrecen respecto a la manera tradicional con que se ha llevado a cabo el análisis de series temporales. Finalmente se comprueba experimentalmente la utilidad de las herramientas implementadas.

2.2 Descripción general de Weka

Weka es un software de código abierto escrito en Java, creado en la Universidad de Waikato en Nueva Zelanda, y publicado bajo la Licencia Pública General de GNU[4]. Cuenta con una colección de algoritmos de aprendizaje automático para tareas de análisis de datos y modelado predictivo, que se aplican a la minería de datos [67]. Además contiene una interfaz gráfica de usuario que facilita el acceso a sus funcionalidades [68] .

Las versiones más recientes de Weka tienden a extender sus funcionalidades a distintas áreas de investigación, en particular con fines docentes e investigativos. En la Universidad Central “Marta Abreu” de Las Villas se utiliza este software como una herramienta para diferentes fines, como es el caso de la impartición de docencia para la

[3] Este software toma su nombre del acrónimo del inglés *Waikato Environment for Knowledge Analysis,* que significa Entorno para Análisis del Conocimiento de la Universidad de Waikato.

[4] GNU es un acrónimo formado a partir de la frase en inglés “*GNU is Not Unix”,* y que significa GNU no es Unix.

asignatura de Inteligencia Artificial. También es usado en grupos de investigación como el de Bioinformática y el de Inteligencia Artificial donde se le han agregado nuevas funcionalidades.

2.3 Paquete de Weka para la predicción de series temporales

Desde versiones posteriores a **Weka-3.7.3** se encuentra disponible un paquete desarrollado por *Pentaho Community* denominado **timeseriesForecasting** el cual está dedicado específicamente a la predicción de series temporales. Dicho paquete constituye una extensión a Weka y se incorpora añadiéndose como una pestaña dentro de la interfaz **Explorer** de Weka, permitiendo el desarrollo de modelos predictivos de los datos, y además su evaluación y visualización de forma clara y precisa, ver ANEXO 1.

El paquete **timeseriesForecasting** utiliza una máquina de aprendizaje automatizado con enfoque de minería de datos para crear los modelos de la serie de tiempo y las predicciones del comportamiento futuro de la serie. Esto se logra transformando los datos en un formato que los métodos tradicionales de aprendizaje automatizado puedan manejar. Para esto se elimina el orden temporal de los datos de entrada individuales, mediante la codificación de las dependencias temporales, adicionándole nuevos campos a los datos. Además, otros campos son calculados y adicionados automáticamente con el propósito de permitirle la modelación de los distintos fenómenos que pueden presentarse en la serie, tales como la tendencia y la estacionalidad.

Posterior a la transformación de los datos, cualquier algoritmo de Weka que implemente regresión puede aprender de dicho modelo. Una elección obvia es aplicar regresión lineal múltiple, pero cualquier método capaz de predecir un atributo continuo puede ser aplicado. Esto incluye los métodos no lineales tales como máquinas de soporte vectorial para regresión, y árboles de decisión con funciones de regresión lineales en las hojas. Para una información completa de las características y uso del paquete de Weka antes mencionado visitar [69].

A pesar de las facilidades que ofrece esta extensión para el análisis de series temporales, estas están enfocadas fundamentalmente en el pronóstico. Weka aún no cuenta con suficientes funciones para el trabajo eficaz con series temporales, especialmente de aquellas que faciliten su clasificación.

2.4 Diseño e implementación de un paquete con herramientas para la clasificación de series temporales en Weka

En este epígrafe se propone la inclusión de extensiones a Weka, las cuales facilitan la clasificación de series temporales. Dichas extensiones han sido agrupadas en un paquete denominado **timeSeriesClassification**, ver ANEXO 9.

Primeramente se implementa, como una función de distancia, la medida elástica DTW descrita en el Capítulo 1. Se propone su uso en el algoritmo kNN, combinada con una cota inferior de DTW, la cual es implementada como un algoritmo de búsqueda de vecinos más cercanos. También se implementa un filtro basado en kNN para la reducción de la numerosidad de conjuntos de datos numerosos, ya que este constituye un problema que se presenta muy frecuentente durante el análisis de series temporales.

La compilación y construcción del paquete **timeSeriesClassification** el cual contiene las extensiones agregadas a Weka fue realizada usando Apache Ant versión **1.9.1** [70], dicho paquete puede cargarse en cualquier versión de Weka posterior a la **3.7.7**. A continuación se describen los pasos que se siguieron para implementar cada una las extensiones, la metodología para extender Weka utilizada es la presentada en [71].

2.4.1 Inclusión en Weka de una función de distancia para series temporales

A pesar de que la mayoría de los algoritmos de aprendizaje basado en instancias utilizan la distancia euclidiana, se considera a DTW como la medida más usada actualmente para la clasificación de series temporales [45, 72]. Como ha sido explicado en el Capítulo 1, DTW supera los problemas que presenta la distancia euclidiana a la hora de medir la diferencia entre dos series temporales, en especial aquellas con desfases en el tiempo. Las ventajas de su uso han sido demostradas en una gran cantidad de artículos científicos [73-75].

Los métodos de minería de datos basados en kNN son altamente sensibles a la función de distancia que utilizan [76]. A pesar de que numerosos algoritmos para la clasificación de series temporales han sido propuestos, incluyendo árboles de decisión [77], redes neuronales [78], redes bayesianas [78] etc.; evaluaciones empíricas en más de 40 conjuntos de datos internacionales han mostrado que el clasificador 1NN-DTW es superior a la mayoría de las técnicas usadas en la clasificación de series temporales [34, 45].

Por todo lo anterior, se propone la inclusión de DTW a Weka como una métrica de distancia. En particular, se sugiere el uso de 1NN-DTW; por ser este el clasificador más usado actualmente para este tipo de series.

La implementación de DTW realizada utiliza la banda Sakoe-Chiba como una restricción global opcional en su cálculo. El ANEXO 2 muestra el diagrama de clases de la función de distancia **DTWDistance** implementada. Para la inclusión en Weka de **DTWDistance** se siguieron los siguientes pasos:

1. Creación de la clase:

Primeramente, se crea una nueva clase con el nombre **DTWDistance** y esta se sitúa en el paquete "*weka.core*". Se ubica en este paquete, que constituye el centro del sistema Weka, debido a que es en él donde tradicionalmente se han implementado las funciones de distancia en Weka.

2. Importación de paquetes:

Deben ser importados los paquetes necesarios, primeramente el paquete obligatorio para cualquier función de distancia: "*weka.core.neighboursearch.PerformanceStats*". Además es necesario importar el paquete "*java.util*" debido a que la nueva función de distancia necesita parámetros de entrada; de ellos, el más importante es el que define el tamaño de la ventana que representa la banda Sakoe-Chiba implementada.

3. Herencia de clases e implementación de interfaces:

Esta función de distancia, a diferencia de otras como **EuclideanDistance** y **ManhattanDistance** que son normalizables y por ello heredan de la clase abstracta **NormalizableDistance**, no es una métrica normalizable. Es por ello que en lugar de heredar de **NormalizableDistance** la clase **DTWDistance** simplemente implementa la interfaz **DistanceFunction**. Otras cuatro interfaces son implementadas: **OptionHandler**, **Serializable** y **Cloneable.** La primera porque **DTWDistance** necesita parámetros de entrada, la segunda y la tercera para facilitar la serialización y la copia de la clase respectivamente.

4. Parámetros de entrada:

Las opciones de entrada que necesita **DTWDistance** son:

- ✓ el rango de los atributos a tener en cuenta en el cálculo de la distancia.
- ✓ el tamaño de la ventana que representa el ancho de la banda de Sakoe-Chiba.

Los valores de cada una de estas opciones son almacenados en la variables *m_AttributeIndices* y *m_WindowSize* respectivamente. La variable *m_AttributeIndices* es de tipo **Range**, clase implementada en Weka que se utiliza para almacenar rangos, y por defecto tiene valor "*first-last"* significando que van a ser preprocesados todos los atributos. El valor por defecto de la variable *m_WindowSize* es 10, que representa un ancho de ventana Sakoe-Chiba igual al 10% de la longitud de las series temporales en el conjunto de datos.

Al utilizar la interfaz **OptionHandler** deben ser implementados los métodos que ella posee (**listOptions**, **setOptions**, **getOptions**). En **listOptions** se construyen las dos opciones de la métrica. El método **setOptions** recibe como parámetro las opciones en forma de arreglo de cadena, y es el encargado de validar cada una de ellas y almacenarlas mediante los métodos **setWarpingWindowSize** y **setAttributeIndices**.

Las opciones válidas son:

-R: especifica el índice de los atributos a utilizar.

-W: valor entero que especifica el tamaño (dado como el porciento de la longitud de las series temporales del conjunto de datos) de la ventana que representa la banda de Sakoe-Chiba a utilizar.

El método **getOptions** devuelve la configuración actual de la métrica en un arreglo de cadena. Esta lista es retornada de forma tal que pueda ser pasada al método **setOptions**, con la estructura: -R, índice de los atributos a utilizar, -W, el tamaño de la ventana.

Por cada una de las opciones declaradas se definieron los métodos para colocar su valor en la variable creada para ello, devolver el valor de esa variable y mostrar información adicional de cada opción (métodos **set**, **get** y **TipTexts**).

5. Métodos implementados de la interfaz DistanceFunction:

distance: Es el método principal de la clase, calcula la distancia DTW entre dos series temporales (instancias) de la base de casos.

getInstances: Devuelve las instancias con las que trabajará la métrica. Estas han sido almacenadas en la variable global de tipo **Instances** denominada *m_Data*.

getAttributeIndices: Devuelve el rango de los atributos usados en el cálculo de la distancia.

getInvertSelection: Devuelve si tiene sentido que la selección de los atributos esté invertida.

6. Método de uso interno de la clase:

- **DistanceDTW**: Método interno que calcula dinámicamente la distancia DTW entre dos series temporales (instancias) usando como restricción global la banda de Sakoe-Chiba, para acelerar la velocidad de los cálculos.

El ANEXO 5 muestra la interfaz gráfica de la función de distancia **DTWDistance** implementada en Weka.

2.4.2 Inclusión en Weka de un algoritmo para la búsqueda de *k* vecinos más cercanos para kNN

El uso de la cota inferior LB_Keogh ha sido adoptado en una gran cantidad de aplicaciones que requieren que los cálculos de DTW se realicen de forma eficiente. Esta adopción ha facilitado el uso de DTW en la minería de conjuntos de datos numerosos, constituidos por series temporales cada vez más extensas.

Teniendo en cuenta lo dicho anteriormente, se propone la inclusión de un algoritmo de búsqueda de vecinos más cercanos denominado **DTWSearch**, basado en el algoritmo de la Figura 2.1 **Algoritmo que utiliza**, para la clasificación de series temporales en Weka. **DTWSearch** está basado en la distancia DTW y utiliza la función LB_Keogh como cota inferior de DTW. De esta forma es posible ahorrar cómputos innecesarios de dicha métrica mejorando significativamente la eficiencia del clasificador.

Figura 2.1 Algoritmo que utiliza LB_Keogh para seleccionar las series más similares a la serie C en el conjunto de datos *Q*.

Algorithm Lower_Bounding_Sequential_Scan(Q,C)

```
1: best_so_far = Inf
2: for all sequences in database C do
3:    LB_dist = lower_bound_distance(C_i, Q) //Cheap lower bound
4:    if LB_dist < best_so_far then
5:       true_dist = DTW(C_i, Q) // Expensive DTW
6:       if true_dist < best_so_far then
7:          best_so_far = true_dist
8:          index_of_best_match = i
```

El ANEXO 3 muestra el diagrama de clases del algoritmo para la búsqueda de los *k* vecinos más cercanos del kNN implementado. Para la inclusión en Weka de **DTWSearch** se han llevado a cabo los siguientes pasos:

1. Creación de la clase:

Primeramente, se crea una nueva clase con el nombre **DTWSearch** y se sitúa en el paquete "*weka.core.neighboursearch*". Se ubica en este paquete porque es en él donde Weka implementa todos sus algoritmos de búsqueda de vecinos más cercanos (como por ejemplo **LinearNNSearch** para la búsqueda mediante fuerza bruta de los *k* vecinos más cercanos de una instancia dada).

2. Importación de paquetes:

Debe ser importado solamente el paquete "*weka.core*" ya que es en este paquete donde se encuentran las clases **Option**, **Instance**, **Instances** y **DTWDistance**; entre otras que el algoritmo utiliza durante la búsqueda de los vecinos más cercanos de una serie dada.

3. Herencia de clases e implementación de interfaces:

LinearNNSearch hereda de la clase abstracta **NearestNeighbourSearch**, todos los algoritmos que realicen la búsqueda de los vecinos más cercanos de una instancia dada deben heredar de esta clase. Por tanto, **DTWSearch** también hereda de **NearestNeighbourSearch**, y no requiere implementar ninguna interface.

4. Parámetros de entrada

Las opciones de entrada que necesita el algoritmo son:

- ✓ la función de distancia utilizada por el algoritmo para la búsqueda de los vecinos más cercanos (debe ser necesariamente **DTWDistance**), y sus parámetros correspondientes.
- ✓ si se desea calcular los estadísticos de desempeño para la búsqueda NN o no.
- ✓ si se desea obviar las instancias idénticas o no.

Los valores de cada una de estas opciones son almacenados en la variables *m_DistanceFunction, m_Stats* y *m_SkipIdentical* respectivamente. La primera opción es

la más importante en el funcionamiento del algoritmo, *m_DistanceFunction* es de tipo **DistanceFunction** y debe estar instanciada con la función de distancia **DTWDistance**.

Las opciones válidas son:

-A: especifica la función de distancia a utilizar.

-P: especifica si se desea calcular los estadísticos de desempeño.

-S: especifica si se desea obviar las instancias idénticas

El método **getOptions** devuelve la configuración actual del algoritmo en un arreglo de cadena. Esta lista es retornada de forma tal que pueda ser pasada al método **setOptions**, con la estructura: -A, **DTWDistance**, -P, *true* o *false* según se desee calcular los estadísticos de desempeño y –S, *true* o *false* según se desee obviar las instancias idénticas.

Por cada una de las opciones declaradas se definieron los métodos para colocar su valor en la variable creada para ello, devolver el valor de esa variable y mostrar información adicional de cada opción (métodos **set**, **get** y **TipTexts**).

5. Métodos redefinidos de la clase abstracta NearestNeighbourSearch:

- **kNearestNeighbours**: Es el método principal de la clase, retorna los vecinos más cercanos de una serie dada. Utiliza la función de distancia **DTWDistance** para el cálculo de la distancia entre dos de tiempo dadas. Calcula la función LB_Keogh y utiliza este valor como cota mínima de DTW. Con esto se evita el cálculo de DTW cuando el valor LB_Keogh, entre la serie dada y una del conjunto de datos, es mayor que el menor valor de distancia encontrado hasta el momento.
- **setDistanceFunction**: Pone la función de distancia en la variable *m_DistanceFunction* forzando a que esta sea **DTWDistance**.

6. Métodos de uso interno de la clase:

- **computeL**: Calcula la serie que representa el límite superior de la envolvente para la serie dada.
- **computeB**: Calcula la serie que representa el límite inferior de la envolvente para la serie dada.
- **computeLB_Keogh**: Calcula el valor de la cota mínima de la distancia DTW entre una serie de tiempo y la envolvente correspondiente a la serie objetivo.

El ANEXO 6 muestra la interfaz gráfica del algoritmo de búsqueda de los *k* vecinos más cercanos **DTWSearch** implementado en Weka.

2.4.3 Inclusión en Weka de un filtro para la reducción de la numerosidad de series temporales

Muchos algoritmos han sido propuestos en el campo de la clasificación de series temporales. Como ya se ha dicho, aquellos basados en vecinos más cercanos (en inglés *one nearest neighbor*) que implementan DTW como función de distancia, son los que mejores resultados ofrecen y por consiguiente son difíciles de mejorar. Sin embargo existe una dificultad significativa en cuanto al tiempo de obtención de los resultados con este tipo de algoritmos; en gran medida producto de la numerosidad de los datos que generalmente conforman las series temporales, su alta dimensionalidad y la necesidad de su constante actualización.

Todo esto, sumado a la demanda de resultados inmediatos, por aplicaciones en tiempo real que los requieren, trae consigo la necesidad de ejecutar la clasificación lo más rápidamente posible. En este sentido, se ha trabajado mucho para acelerar la velocidad en los cálculos de DTW. Numerosos avances se han dado, no obstante existe un claro límite en cuanto a estas mejoras; incluso se ha sugerido la existencia de un límite asintótico en cuanto a qué tanto se podría mejorar la eficiencia de DTW [79].

La idea de reducir la numerosidad de los datos ofrece ventajas adicionales [80]. Es bien conocido que, si se escogen con cuidado las instancias que va a descartar un clasificador, esto puede reducir significativamente el tiempo de ejecución de la clasificación, a la vez que se mantiene la efectividad del clasificador (en muchos casos incluso los resultados obtenidos son mejores luego de efectuada la reducción).

Investigaciones recientes han mostrado que la utilización de DTW con restricciones óptimas, unido al uso de algoritmos que reducen convenientemente la numerosidad de los datos, dan como resultado conjuntos de datos muy compactos y con poca o ninguna pérdida de precisión en la clasificación de series temporales [34]. Las técnicas para la reducción de la numerosidad existentes incluyen: reducción aleatoria, condensado, edición de los datos y muchas otras más.

Debido a su importancia, se propone la inclusión a Weka de una herramienta de preprocesamiento de datos para la reducción de la numerosidad de series temporales. Dicha herramienta se implementa como un filtro, el cual está orientado a la eliminación de aquellas series que menos aporten a la clasificación, teniendo siempre en cuenta la interdependencia entre el atributo clase y los valores de los demás atributos de la serie.

El algoritmo implementado en Weka, denominado Rank Based Reduction (RBR) toma como base los algoritmos de [81], uno de los más referenciados en esta área. La idea intuitiva en que se inspira el algoritmo es simple: si la eliminación de una instancia p, en

un conjunto de datos *S,* no produce que otras instancias en *S* sean mal clasificadas entonces p puede ser extraída de *S* sin que por ello se afecte la clasificación.

RBR funciona en dos etapas, las cuales llamaremos "jerarquización" y "desempate". Primeramente se asigna una jerarquía a cada una de las series temporales (instancias), que van a ser clasificadas con 1NN-DTWDistance, según su aporte a la clasificación de todo el conjunto de datos. Una vez definidas las clases que van a ser reducidas y el porcentaje que se desea reducir de las mismas, se aplican ambas etapas del algoritmo. Finalmente se obtiene un conjunto de datos reducido el cual contiene aquellas series con mayor valor de jerarquía durante cada etapa.

Durante la "jerarquización", se comienza con la eliminación de todas aquellas series duplicadas (si existen), pues estas no le aportan información nueva al clasificador. Luego se aplica 1NN-DTW en todo el conjunto de datos, asignándole una menor jerarquía a aquellas series que ofrecen información "ruidosa" para la clasificación. Esto es debido a que estas series generalmente afectan la clasificación de otras series cercanas. Para cada serie se le asigna un valor de jerarquía según la ecuación (2.1):

$$jerarquía(x) = \sum_j \begin{cases} 1 & si\ clase(x) = clase(x_j) \\ -2 & en\ otro\ caso \end{cases} \qquad (2.1)$$

Donde x_j es la serie que tiene a x como su vecino más cercano. Por consiguiente, se le asigna mayor jerarquía a aquellas series que más aportan en la clasificación del resto, y aquellas que peor lo hacen obtienen valores negativos.

Si dos series tienen la misma jerarquía, entonces el empate se rompe asignándoles diferentes prioridades; esta es la etapa denominada "desempate". La prioridad de una serie *x* es calculada según la siguiente ecuación (2.2):

$$prioridad(x) = \sum_j \left(\frac{1}{\left(\text{DTWDistance}(x,\ x_j)\right)^2} \right) \qquad (2.2)$$

Donde x_j es la serie que tiene a x como su vecino más cercano y $\text{DTWDistance}(x,\ x_j)$ representa la distancia DTW entre x y x_j. El supuesto en este caso es que si una serie está demasiado alejada de su vecino más cercano, entonces este ejerce una menor influencia en la clasificación de la serie vecina. Luego, si dos series tienen la misma jerarquía,

aquella con la menor prioridad será descartada primero. Notar que, gracias a que en la primera etapa se han eliminado las instancias duplicadas, se puede asegurar que el denominador de la fracción será distinto de cero.

El ANEXO 4 muestra el diagrama de clases del filtro para la reducción de la numerosidad implementado. Para la inclusión en Weka de **NumerosityReduction**, basado en el algoritmo RBR propuesto, se han llevado a cabo los siguientes pasos:

1. Creación de la clase:

Primeramente se crea la clase con el nombre **NumerosityReduction** y se sitúa en el paquete "*weka.filters.supervised.instance*". Se ubica dentro de este paquete porque su principal propósito es filtrar las series temporales (instancias) del conjunto de datos, para lo cual se utiliza la información que ofrece el atributo clase.

2. Importación de paquetes:

Deben ser importados los paquetes necesarios, primeramente los dos obligatorios para cualquier filtro: "*weka.filters*" y "*weka.core*". Además es necesario importar el paquete "*java.util*" debido a que el filtro necesita parámetros de entrada y de estructuras de datos, así como el paquete "*weka.classifiers.lazy.IBk*" para usar este clasificador durante el proceso de jerarquización de los datos.

3. Herencia de clases e implementación de interfaces:

RBR hereda de la clase **Filter** e implementa dos interfaces: **OptionHandler** y **SupervisedFilter**. La primera porque se necesitan parámetros de entrada y la segunda debido a que es un filtro supervisado. Esta última se usa solamente como indicador, ya que es una interfaz vacía.

4. Parámetros de entrada:

Las opciones de entrada que necesita el filtro son: índice de cada una de las clases a reducir, la función de distancia a utilizar, y el porciento de la clase (o de las clases) que va a ser reducido en el conjunto de datos. Los valores de cada una de las opciones son almacenados en la variables *m_ClassesToReduce*, *m_DistanceFunction* y *m_Percent* respectivamente. La variable *m_ClassesToReduce* es de tipo *Range (*por defecto tiene valor *first-last),* variable *m_DistanceFunction* es de tipo **DistanceFunction** (por defecto es instanciada con **DTWDistance**) y el valor inicial de *m_Percent* corresponde a un 10% de reducción.

La implementación de los métodos de la interfaz **OptionHandler** se realiza de la misma forma a como se explicó anteriormente en la incorporación de la distancia **DTWDistance**. Las opciones validas son:

-R: especifica el índice de las clases que se van a reducir.

-D: especifica la función de distancia que va a utilizar el filtro para hacer la reducción.

-P: especifica el porciento de la(s) clase(s) que se reducirá(n).

Por cada una de las opciones declaradas se definieron los métodos para colocar su valor en la variable creada para ello, devolver el valor de esa variable y mostrar información adicional de cada opción (**set**, **get** y **TipTexts**).

5. Creación de clases internas:

El filtro implementa una clase interna denominada **InstancesList** la cual se crea con el objetivo de almacenar objetos de este tipo en la lista de jerarquías y poder ordenarla más fácilmente. La clase contiene las siguientes variables:

- *instance*: variable de tipo **Instance**, contiene una serie de tiempo a jerarquizar.
- *nearestNeighbors*: contiene los vecinos más cercanos (**Instances**) de *instance*, los cuales han sido calculados mediante 1NN-DTW.
- *havingAsItsNN*: contiene las instancias (**Instances**) que tienen a cada instancia como su vecino más cercano.
- *rank*: representa el valor jerárquico de *instance* para el conjunto de datos a reducir, calculado según el algoritmo RBR.

6. Métodos redefinidos de la clase abstracta Filter:

- **setInputFormat**: mediante este método se fija el formato de entrada de los datos. Primeramente se llama al método **setInputFormat** de la clase **Filter** y después se inicializa la variable *m_ClassesToReduce*.
- **batchFinished**: es llamado después que se ha terminado de entrar todo el lote de datos. Como los datos deben haber sido entrados completamente para poder comenzar a filtrar es aquí donde se realiza este proceso. La llamada al método privado **rankBasedReduction** pasándole como parámetro el conjunto de datos con **getInputFormat** realiza el filtrado, devolviendo las series ordenadas según su jerarquía en la variable global *m_InstancesRankedPriorities*. A continuación se adicionan a la cola de salida del filtro las series resultantes según el porciento de filtrado especificado, esto se realiza mediante el método **push** de la clase **Filter**. Este

método también limpia el espacio de memoria de entrada y la variable *m_NewBatch* recibe Verdadero nuevamente, porque se ha terminado el lote de entrada.

7. Principales métodos definidos por la clase:

- **insertInstanceList**: Inserta un objeto de tipo **InstancesList** en la lista según el índice de jerarquía de la serie correspondiente, la inserción se realiza utilizando búsqueda binaria para favorecer su eficiencia.
- **removeDuplicateInstances**: Elimina las instancias duplicadas del conjunto de datos.
- **rankInstanceList**: Asigna un valor de jerarquía a cada instancia del conjunto de datos según la ecuación de jerarquía definida anteriormente.
- **reRankInstanceList**: Este método rompe el empate de las series con igual jerarquía asignándole diferentes prioridades según la ecuación definida anteriormente.
- **nearestNeighbors**: Determina los vecinos más cercanos usando el método **kNearestNeighbours** de la clase **IBk**, con la función de distancia definida.
- **rankBasedReduction**: Es el método principal, implementa el algoritmo RBR propuesto.
- **main**: Fue implementado de la misma manera en que deben hacerlo todos los filtros: llamando a los métodos **filterFile** y **batchFilterFile** de la clase **Filter**.
- **globalInfo**: Retorna una breve descripción del funcionamiento del filtro.

El ANEXO 7 muestra la interfaz gráfica del filtro **NumerosityReduction** implementado en Weka.

2.5 Resultados experimentales

En este epígrafe se realiza una validación experimental y estadística de las extensiones implementadas para la clasificación de series temporales en Weka, comparándolas a su vez con los métodos tradicionales. Las comparaciones se efectúan utilizando varios de los estadísticos obtenidos a partir de los experimentos. Los conjuntos de datos utilizados se encuentran disponibles en [64] y proceden de dominios diversos, garantizando así la fiabilidad de los experimentos. Para el diseño de los experimentos se utiliza el propuesto en [82].

2.5.1 Diseño de los experimentos

A partir de los experimentos realizados se desea conocer el desempeño de la nueva función de distancia **DTWDistance** implementada en Weka. Dicha función de distancia va a ser utilizada por el algoritmo kNN para la clasificación de series temporales. Se lleva a cabo una comparación entre dicha métrica y la función de distancia tradicional **EuclideanDistance** de Weka. En todos los experimentos, el clasificador kNN utiliza un vecino más cercano (k = 1), y la función de distancia **DTWDistance** usa la banda Sakoe-Chiba como restricción global de DTW.

Además, se desea comprobar la eficacia del uso del algoritmo **DTWSearch** (el cual usa la función LB_Keogh como cota mínima de DTW), implementado para hacer más eficiente el funcionamiento del algoritmo kNN cuando este usa como función de distancia **DTWDistance**. La validación de **DTWSearch** se realiza sumando la cantidad de cálculos innecesarios de DTW que se evitan con su uso.

También se evalúa la eficiencia del filtro **NumerosityReduction**, para la reducción de la numerosidad de series temporales. Esto se realiza evaluando el desempeño del algoritmo kNN en conjuntos de datos cuya numerosidad se ha reducido con dicho filtro.

Las variables de comprobación utilizadas en los experimentos son de tipo cuantitativo, ya que se van a manejar los principales estadísticos descriptivos derivados de la aplicación de un algoritmo de clasificación de series temporales. La medida utilizada para caracterizar el desempeño de la clasificación es el porciento de las series correctamente clasificadas.

2.5.2 Caracterización de los conjuntos de datos

Para los experimentos se utilizaron diez conjuntos de datos los cuales han sido usados tradicionalmente por los investigadores en el análisis de series temporales, cada uno está provisto de su respectivo conjunto de entrenamiento para el aprendizaje, y de control para las comprobaciones. La Tabla 2.1 muestra algunas de las características fundamentales de dichos conjuntos de datos. Como se puede apreciar, se incluyen en esta selección los problemas ECG, Gun Point y Fifty Words, descritos en el Capítulo 1.

Tabla 2.1 Características de los conjuntos de datos utilizados en los experimentos.

Nombre del conjunto de datos	Número de clases	Tamaño del conjunto de entrenamiento	Tamaño del conjunto de control	Longitud de la serie de tiempo
Gun Point	2	50	150	150
ECG200	2	100	100	96
Yoga	2	300	3000	426
Coffe	2	28	28	286
Trace	4	100	100	275
Waffer	2	1000	6174	152
Fifty Words	50	450	455	270
Two Patterns	4	1000	4000	128
Fish	7	175	175	463
CBF	3	30	900	128

Se seleccionaron estos diez conjuntos de datos en particular ya que cada uno describe un problema perteneciente al dominio de las series temporales con características diferentes (número de clases, tamaño de los conjuntos de entrenamiento y control, longitud de la serie etc.). Además, estos incluyen otras características interesantes tales como la no periodicidad, irregularidad, caos, antisimetría y presencia de ruido. Sus atributos son también de diversa naturaleza: algunos representan signos, fotogramas, colores, olores, patrones etc.

2.5.3 Validación de la función de distancia implementada

A continuación se va evaluar el desempeño de la función de distancia **DTWDistance** implementada en Weka. La Tabla 2.2 muestra los resultados obtenidos después de la aplicación del algoritmo kNN con esta función de distancia en los diez conjuntos de datos seleccionados para los experimentos.

Tabla 2.2 Comparación entre los porcientos de clasificación correcta obtenidos con **1NN-EuclidianDistance**, **1NN-DTWDistance** (con (*r*) donde *r* indica el ancho de ventana con el que mejores resultados se obtuvo), y **1NN-FullDTWDistance** (*100*), o sea DTW sin restricciones.

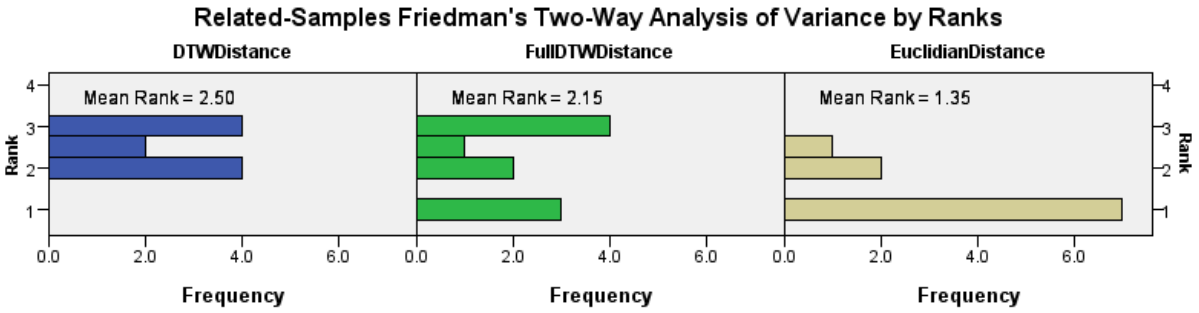
Nombre del conjunto de datos	1NN-EuclidianDistance	1NN-DTWDistance con ancho de ventana (*r*), *r* = % de la longitud de la serie	1NN-FullDTWDistance *r* = 100
Gun Point	91.3333 %	**92 %** (1)	90.6667 %
ECG200	87 %	**87 %** (0)	76 %
Yoga	83.0333 %	**84.1667 %** (2)	83.6 %
Coffe	75 %	**82.1429 % (3)**	82.1429 %
Trace	76 %	99 % (3)	**100 %**
Waffer	99.5457 %	**99.562 %** (1)	97.9883 %
Fifty Words	63.0769 %	**76.4835 %** (6)	69.011 %
Two Patterns	90.675 %	99.725 % (4)	**100 %**
Fish	78.2857 %	82.8571 % (4)	**83.4286 %**
CBF	85.2222 %	99.4444 % (11)	**99.6667 %**

Se puede apreciar que para todos los conjuntos de datos, el porciento de series bien clasificadas usando 1NN con DTW como función de distancia (tanto con un ancho de ventana r óptimo como sin restricción global de ventana), es superior (o igual en el peor de los casos) al porciento obtenido usando la distancia euclidiana de Weka. Ha continuación van a ser realizadas algunas comprobaciones estadísticas (usando SPSS versión 21) con el objetivo de confirmar lo dicho anteriormente.

Se desea comprobar primeramente que existen diferencias significativas entre los porcientos de series bien clasificadas usando **1NN-EuclidianDistance**, **1NN-DTWDistance** y **1NN-FullDTWDistance**.

Dado que la cantidad de conjuntos de entrenamiento que conforman la muestra no es lo suficientemente grande como para probar normalidad y aplicar pruebas paramétricas, y que la comparación se va a realizar sobre varios algoritmos simultáneamente [47], entonces se va a utilizar la prueba de Friedman [48]. La Figura 2.2 muestra los resultados obtenidos con esta prueba.

Figura 2.2 Resultados del análisis de varianzas de Friedman para los porcientos de clasificación de 1NN con **EuclidianDistance**, **DTWDistance** y **FullDTWDistance**.



Total N	10
Test Statistic	7.316
Degrees of Freedom	2
Asymptotic Sig. (2-sided test)	.026

El análisis de varianza, con un grado de significación de ($\alpha = 0.05$) da como resultado, para las funciones de distancia **DTWDistance** y **FullDTWDistance**, un rango medio de varianza de 2.50 y 2.15 respectivamente, mientras que para **EuclidianDistance** este tiene un valor de 1.35 (significativamente menor a los dos primeros). Luego, la significación obtenida es de 0.026, y por lo tanto se rechaza la hipótesis nula: las distribuciones de los porcientos de clasificación correcta no son equivalentes entre estas funciones de distancia, Figura 2.3.

Figura 2.3 Se rechaza la hipótesis nula: según la prueba de Friedman las distribuciones de **DTWDistance**, **FullDTWDistance** y **EuclidianDistance** no son las mismas.

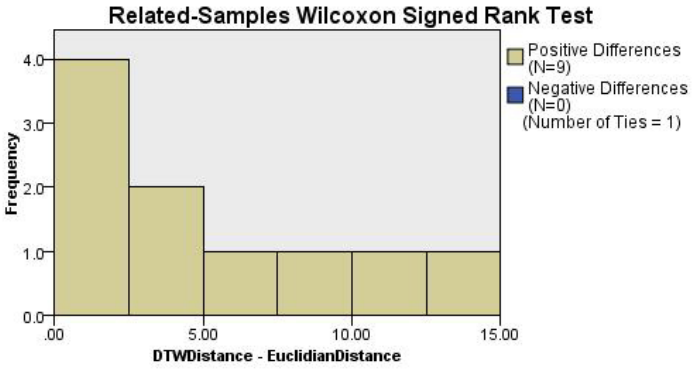
Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distributions of DTWDistance, FullDTWDistance and EuclidianDistance are the same.	Related-Samples Friedman's Two-Way Analysis of Variance by Ranks	.026	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Para comprobar más claramente las diferencias entre **DTWDistance** y **EuclidianDistance**, se va a realizar entre ellas una prueba no paramétrica de signos de Wilcoxon [46]. La Figura 2.4 muestra los resultados obtenidos.

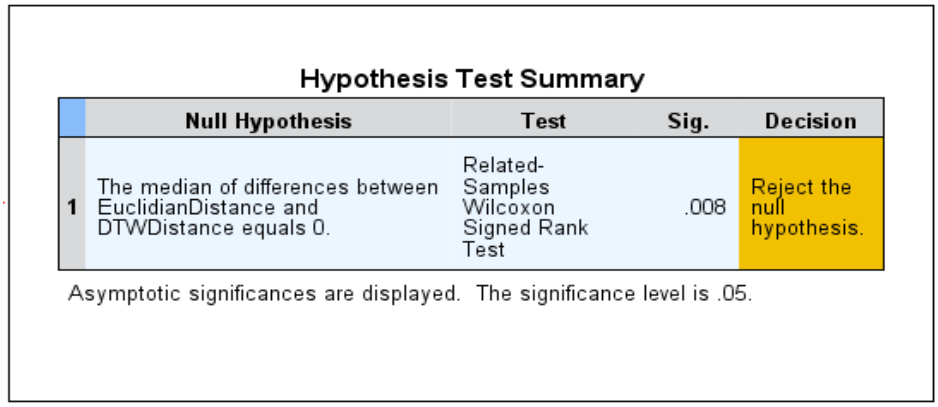
Figura 2.4 Resultados de la prueba no paramétrica de signos de Wilcoxon para los porcientos de clasificación de 1NN con **EuclidianDistance**, **DTWDistance** y **FullDTWDistance**.



Total N	10
Test Statistic	45.000
Standard Error	8.441
Standardized Test Statistic	2.666
Asymptotic Sig. (2-sided test)	.008

Como se puede apreciar, el número de diferencias positivas entre **EuclidianDistance** y **DTWDistance** es de 9, el número de diferencias negativas es igual a cero, y hubo un solo empate. Luego, la significación obtenida es de 0.08, y por lo tanto se rechaza la hipótesis nula: la media de las diferencias de los porcientos de clasificación correcta no son equivalentes entre estas funciones de distancia, Figura 2.5.

Figura 2.5 Se rechaza la hipótesis nula: según la prueba no paramétrica de signos de Wilcoxon, la media de las diferencias entre los porcientos de clasificación de 1NN con **DTWDistance** y **EuclidianDistance** es diferente de cero.



Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The median of differences between EuclidianDistance and DTWDistance equals 0.	Related-Samples Wilcoxon Signed Rank Test	.008	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

En resumen, los resultados estadísticos obtenidos para los 10 conjuntos de datos seleccionados corroboran las afirmaciones realizadas en [45]. Los porcientos de clasificación correcta alcanzados con **DTWDistance** fueron muy superiores (e iguales en el peor de los casos) a los conseguidos con **EuclidianDistance** para los conjuntos de datos que se utilizaron en los experimentos. La función de distancia **DTWDistance** (con restricción global o sin ella) implementada en Weka, es superior a la función de distancia euclidiana **EuclidianDistance** de Weka.

2.5.4 Validación del algoritmo para la búsqueda de los *k* vecinos más cercanos implementado

Para probar la eficacia de **DTWSearch** como algoritmo para la búsqueda de los *k* vecinos más cercanos, se contabiliza la cantidad de cálculos de DTW que se podan durante la ejecución de **1NN-DTWDistance** usando **DTWDistance** como función de distancia. En todos los casos las comparaciones fueron realizadas utilizando el ancho de ventana óptimo de la Tabla 2.2. A continuación se muestran los resultados obtenidos.

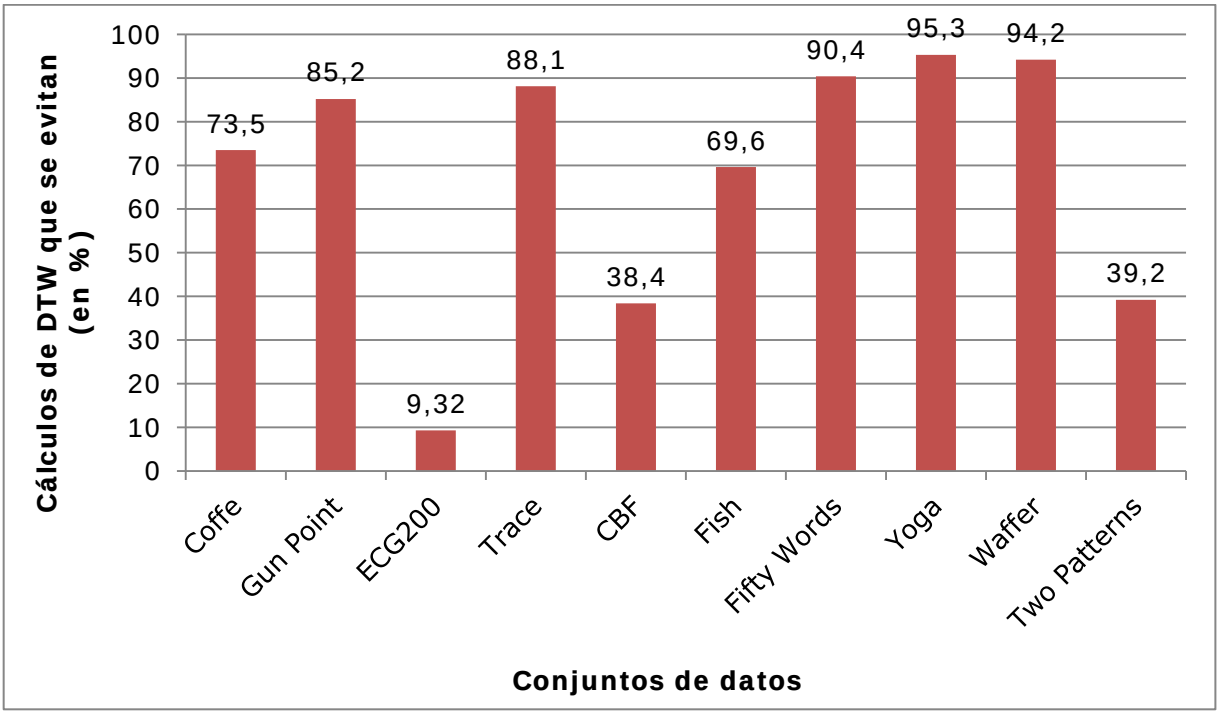
Tabla 2.3 Cantidad de cálculos de DTW que se realizan en cada conjunto de datos usando **LinnearNNSearch** y **DTWSearch** como algoritmos de búsqueda de *k* vecinos más cercanos para 1NN.

Nombre del conjunto de datos	Cálculos de DTW usando LinearNNSearch	Cálculos de DTW usando DTWSearch
Coffe	784	208
Gun Point	7500	1109
ECG200	10000	9068
Trace	10000	1187
CBF	27000	16645
Fish	30625	9301
Fifty Words	204750	19620
Yoga	900000	42132
Waffer	6174000	360688
Two Patterns	40000000	24310997

Como se puede apreciar en la Tabla 2.3, la cantidad de cálculos de DTW que se podan son significativos (para los conjuntos de datos Waffer y Two Patterns andan en el orden de los millones). **DTWSearch** permite entonces que el algoritmo **1NN-DTWDistance** tenga un costo computacional temporal menor, ya que evita numerosos cálculos innecesarios de DTW.

Para ver esto más claramente, la Figura 2.6 muestra gráficamente el porciento de la cantidad de la cantidad de cálculos de DTW que se evitan en cada uno de los diez conjuntos de datos, cuando se usa **DTWSearch** como algoritmo de búsqueda de los *k* vecinos más cercanos, en lugar de usar el algoritmo **LinnearNNSearch** con **DTWDistance**.

Figura 2.6 Porciento de la cantidad de cálculos de DTW que se evitan cuando se usa **DTWSearch** como algoritmo de búsqueda de los *k* vecinos más cercanos para 1NN, en lugar de **LinearNNSearch** con **DTWDistance**.



Como se puede apreciar en la figura anterior, el número de cálculos de DTW que se evitan usando **DTWSearch** es significativo, y más aún en aquellos conjuntos de datos cuyos conjuntos de entrenamiento y de control son numerosos (Yoga, Waffer, Two Patterns), pues en estos la cantidad de cálculos que se evitan son mucho mayores. De ahí la utilidad que tiene el uso de **DTWSearch** como algoritmo de búsqueda de vecinos más cercanos, especialmente cuando el conjunto de datos conformado por series temporales que se va a clasificar es significativamente numeroso.

2.5.5 Validación del filtro implementado

A continuación se desea comprobar la efectividad del filtro **NumerosityReduction** implementado en Weka para la reducción de la numerosidad de series temporales. Como se puede apreciar en la Tabla 2.1, los conjuntos de datos Two Patterns y Waffer tienen conjuntos de entrenamiento conformados por 1000 series cada uno (una cantidad significativamente mayor al resto), por esta causa, ambos van a ser seleccionados para evaluar la efectividad del filtro. Además se incluye en estos experimentos el conjunto ECG como ejemplo de los resultados que pueden ser obtenidos usando el filtro en un conjunto de datos menos numeroso.

En todos los casos, **NumerosityReduction** va a ser aplicado usando la distancia **DTWDistance** como función de distancia del algoritmo kNN con un ancho de ventana igual al 10% de la longitud de la serie (la cual no es necesariamente la longitud óptima). La Tabla 2.4 muestra los resultados obtenidos después de la aplicación del filtro con reducciones porcentuales ascendentes en los tres conjuntos de datos.

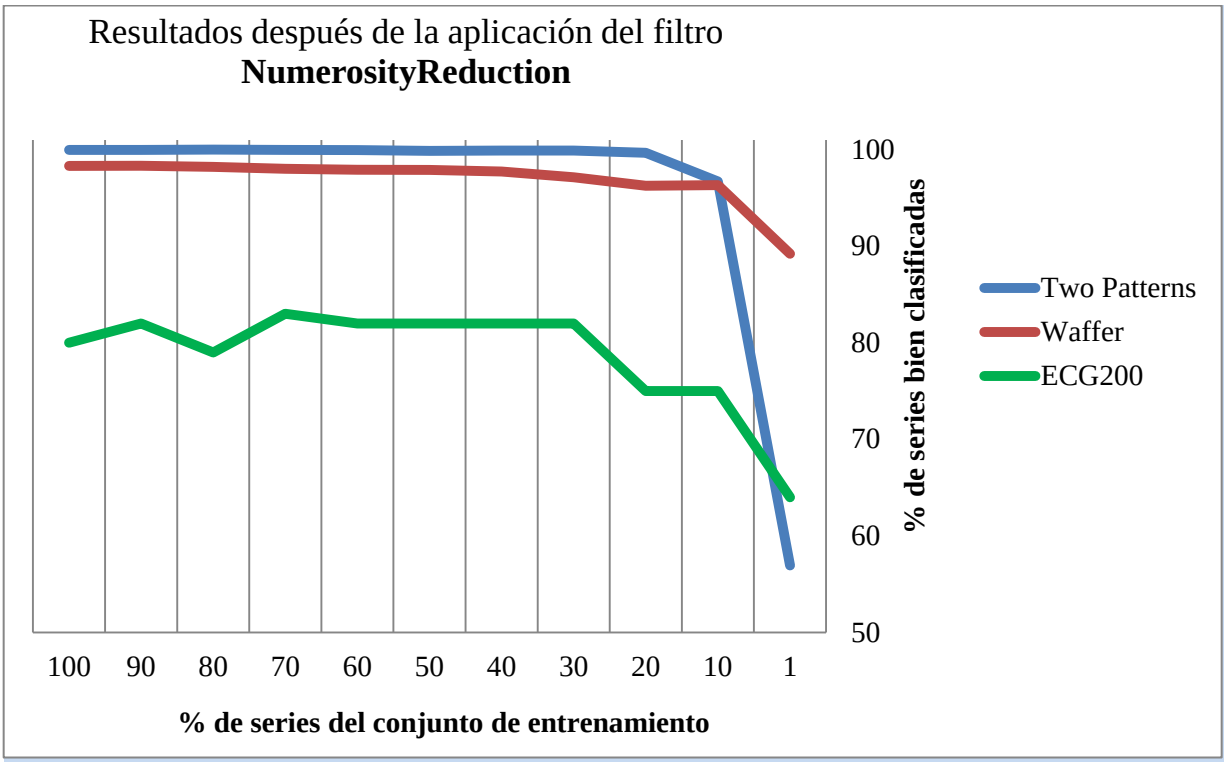
Tabla 2.4 Porciento de series bien clasificadas según el porciento de series reducidas en el conjunto de entrenamiento para los conjuntos de datos Two Patterns, Waffer y ECG200.

% del conjunto de entrenamiento que se reduce	% de series bien clasificadas para el conjunto de datos Two Patterns	% de series bien clasificadas para el conjunto de datos Waffer	% de series bien clasificadas para el conjunto de datos ECG200
0	99.975	98.313	80
10	99.975	**98.329**	82
20	**100**	98.215	79
30	99.975	98.021	**83**
40	99.95	97.9234	82
50	99.875	97.9072	82
60	99.9	97.7287	82
70	99.9	97.1447	82
80	99.675	96.2524	75
90	96.7	96.3173	75
99	56.95	89.2116	64

Como se puede apreciar, en el caso de Two Patterns los mejores porcientos de series bien clasificadas en el conjunto de control se obtienen después de haber reducido el conjunto de entrenamiento un 20%, mientras que para Waffer y ECG200 los mejores resultados se obtienen con una reducción del conjunto de entrenamiento de 10% y 30% respectivamente. Esto sucede debido a que el filtro lo que hace no es simplemente eliminar series de los conjuntos de entrenamiento según lo deseado, sino que elimina las series que menos aportan a la clasificación. Gracias a eso, son eliminadas precisamente aquellas series que suelen ser ruidosas y que por lo tanto solamente dificultan el aprendizaje.

La Figura 2.7 permite tener una idea más precisa de los resultados obtenidos con el uso del filtro para los tres conjuntos de datos utilizados en el experimento. Como se puede apreciar, para el caso de los conjuntos de datos Two Patterns y Waffer, los porcientos de clasificación del conjunto de prueba no varían significativamente hasta que se reduce un 90% del conjunto de control. Esto demuestra que el filtro implementado es sumamente efectivo para la reducción de la numerosidad de estos dos conjuntos de datos. En el caso de ECG200, es efectiva la reducción hasta de un 70% del conjunto de entrenamiento, con lo cual se puede afirmar que el filtro no solo resulta ser eficaz en la reducción de conjuntos de datos con gran numerosidad, sino que también parece ser efectivo para otros no tan numerosos.

Figura 2.7 Porciento de series bien clasificadas según el porciento de series reducidas en el conjunto de entrenamiento para los conjuntos de datos Two Patterns, Waffer y ECG200.



2.6 Conclusiones del capítulo

La herramienta de aprendizaje automatizado Weka no incluye los algoritmos necesarios para efectuar la tarea de clasificación en el dominio de las series temporales. Atendiendo a esto se implementó en el marco de dicho ambiente:

- ✓ Una función de distancia basada en DTW la cual se comporta significativamente superior a la distancia euclidiana al efectuarse su validación en el contexto del algoritmo 1NN.
- ✓ Un algoritmo para efectuar la búsqueda de los *k* vecinos más cercanos, el cual toma ventaja de la cota inferior LB_Keogh, con resultados que muestran de forma experimental una reducción considerable del costo computacional.
- ✓ Un filtro supervisado para la reducción de la numerosidad, situación muy común en el campo de las series temporales, el cual efectúa una eliminación inteligente de aquellas instancias que entorpecen la clasificación. De esta forma logra mantener, por más tiempo, la calidad de la clasificación durante el proceso de reducción.

3 PRONÓSTICO DE PRECIPITACIONES A PARTIR DEL MODELO GFS[5]

3.1 Introducción al capítulo

El pronóstico de precipitaciones resulta ser una tarea extremadamente difícil, numerosos modelos han sido aplicados en este sentido, pero los resultados aún distan mucho de ser confiables. El presente capítulo aborda el problema del pronóstico de precipitaciones, en especial aplicando técnicas de minería de datos a las salidas numéricas del modelo global GFS. Los resultados obtenidos serán comparados con los valores de precipitaciones reales medidos por el Instituto de Meteorología de la provincia de Villa Clara.

3.2 El problema del pronóstico de precipitaciones

El pronóstico de precipitaciones resulta ser una tarea extremadamente difícil y continúa siendo un reto para los científicos dedicados a las investigaciones atmosféricas en todo el mundo. En los últimos años, los modelos numéricos han contribuido a mejorar notablemente estos pronósticos. Sin embargo, aún los resultados distan mucho de satisfacer la necesidad real que se tiene de aprovechar al máximo el servicio meteorológico, en función de incrementar la eficiencia de las economías nacionales.

Los modelos actualmente implementados en los grandes centros mundiales, como son el GFS de los Estados Unidos y el del Centro Europeo de Pronósticos a Mediano Plazo radicado en Londres, no resuelven las necesidades que se plantean en materia de pronóstico de procesos cuya escala se encuentra por debajo de la escala sinóptica. Entre ellos los que identifican las lluvias cálidas y las tormentas convectivas de verano, sobre todo cuando son consecuencia de circulaciones locales de viento, como son las brisas, típicas de las zonas costeras.

---

[5] Acrónimo de *Global Forecast System*, en castellano Sistema Global de Predicción. Es un modelo numérico de predicción meteorológica que se actualiza cuatro veces al día con predicciones que alcanzan hasta los 16 días, aunque su resolución espacial y temporal decrece con el tiempo. Además de ser uno de los modelos más utilizados para la predicción meteorológica a mediano plazo y a escala sinóptica, el GFS es el único de los modelos con cobertura global cuya producción de salidas de predicción están disponibles gratuitamente y bajo dominio público a través de Internet.

Un valioso intento dirigido a resolver esta situación ha constituido el desarrollo de modelos no hidrostáticos, empleados en áreas limitadas con resoluciones que frecuentemente se encuentran por debajo de los 15 kilómetros. Entre los más utilizados en el mundo se encuentran el MM5 (Modelo Mesoescalar de Quinta Generación) y más recientemente el WRF (acrónimo del inglés *Weather Research & Forecast*). Estos modelos logran considerar más detalladamente las características físico geográficas de las regiones y sus resultados generalmente son muy superiores a los de los modelos de mediana y baja resolución. La mayor limitante en el empleo de estos modelos radica en que se necesitan ordenadores con elevadas capacidades de cómputo; de manera que se garantice que la información esté disponible justo cuando se le necesita para las labores de pronóstico y prevención.

Propuesta del Instituto de Meteorología de Santa Clara

En Cuba los departamentos de meteorología realizan un uso considerable de las salidas numéricas de los modelos de pronóstico globales. La técnica más utilizada para ello es la de reducción de escala, que consiste en tomar las salidas de un modelo de circulación general y anidarlo en un modelo de circulación general regional con mayor resolución. También se realizan transformaciones estadísticas de las salidas de dichos modelos con el propósito de realizar pronósticos estadísticos directamente, a partir de determinadas variables de interés seleccionadas previamente.

El Instituto de Meteorología de la provincia de Villa Clara ha elaborado un método de pronóstico de precipitaciones que no contempla el anidamiento de modelos no hidrostáticos de alta resolución. Dicho método consiste en pronosticar el campo de precipitaciones directamente a partir de las salidas numéricas del modelo global de predicción meteorológica GFS, con una resolución de 55 kilómetros, mejorada hasta 10 kilómetros con ayuda de un modelo digital del terreno de 10 kilómetros de resolución y un método de reducción de escala aplicado al campo de viento geostrófico del modelo global, que permitió pronosticar dirección y fuerza del viento para cada estación meteorológica de la red nacional de mediciones meteorológicas del país. Empleando paralelamente para ello métodos del análisis matemático que ajustan las mallas a la resolución seleccionada, Figura 3.1 y Figura 3.2.

Figura 3.1 Malla irregular obtenida al sustituir los datos pronosticados por el Modelo Global GFS por los pronosticados por el modelo estadístico para la red de estaciones de Cuba.

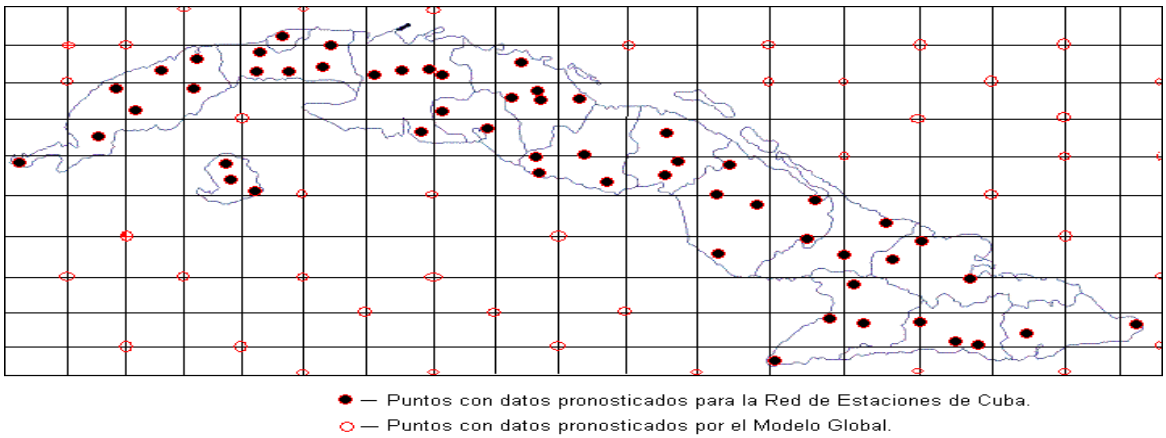
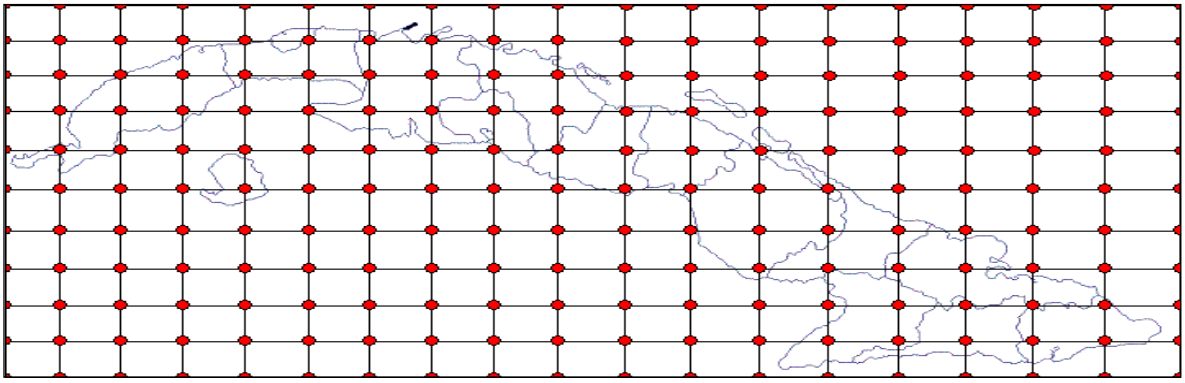


Figura 3.2 Malla regular obtenida con ayuda del método de análisis matemático objetivo "Inverso del cuadrado de la distancia" a partir de la malla mostrada en la Figura 3.1.



La idea de construir un nuevo campo de viento en superficie a partir de un modelo estadístico parte de la hipótesis de que las propiedades estadísticas de estos procesos pueden deducirse del conocimiento de las variables ya resueltas por el modelo numérico. En este caso se emplea como variable predictora, entre otras, el campo pronosticado de presión al nivel medio del mar resuelto por el modelo GFS.

El método de pronóstico de precipitaciones propuesto por el Instituto de Meteorología de Santa Clara se basa en el criterio de que la convergencia del viento en niveles bajos, provocada por la circulación local de brisas y su interacción con el flujo de escala sinóptica, juega un papel decisivo en la formación de precipitaciones en Cuba.

Después de aplicar el método a los datos recogidos por las 68 estaciones meteorológicas del país, el Instituto de Meteorología de la provincia de Villa Clara arribó a las siguientes conclusiones:

- ✓ La verificación parcial arrojó que el modelo logra altos niveles de detección del evento "lluvia", fundamentalmente en el pronóstico para los plazos diurnos 0–12 y 24–36 horas del período lluvioso.
- ✓ Para los plazos de pronóstico 0–12 y 24–36 horas el modelo sobreestima la ocurrencia del evento "lluvia", principalmente en el período poco lluvioso.
- ✓ La efectividad general del modelo en el pronóstico de precipitaciones en 12 horas es de 74% en el período lluvioso y 72% en el poco lluvioso, pero en los plazos de pronóstico 0–12 y 24–36 horas del período lluvioso alcanza una efectividad de 81 y 78% respectivamente.
- ✓ En cuanto al pronóstico cuantitativo de precipitaciones el modelo es altamente confiable para acumulados pronosticados por debajo de los seis milímetros.
- ✓ El modelo subestima la ocurrencia de eventos de lluvias intensas, por ejemplo, más de 50 milímetros en 12 horas.

3.3 Modelación computacional del pronóstico de precipitaciones en Weka

Teniendo en cuenta las deficiencias del modelo de pronóstico de precipitaciones usado actualmente por el Instituto de Meteorología de Santa Clara se plantea la tarea de realizar una modelación del problema utilizando técnicas de aprendizaje automatizado. El objetivo primordial consiste en evaluar su desempeño, partiendo de los resultados obtenidos con los modelos numéricos de pronóstico usados actualmente.

La red de estaciones del sistema meteorológico nacional cuenta con 68 equipos capaces de generar las variables del modelo global GFS. Dichos equipos están distribuidos a todo lo largo y ancho del archipiélago cubano, como se muestra en la Figura 3.1.

Para nuestro análisis se utilizaron dichas variables de salida, las cuales se encuentran disponibles desde el mes de mayo de 2009 hasta octubre de 2012. El plazo de pronóstico es de hasta 48 horas, y comienza a medirse desde las 6*am*.

Para la evaluación de los resultados se emplean diversos índices estadísticos, los cuales son obtenidos a partir de la comparación entre las precipitaciones reales ocurridas, medidas por las estaciones meteorológicas nacionales en intervalos de cada seis horas, y la lluvia pronosticada por el modelo global para igual período de tiempo. Debido a deficiencias técnicas e incumplimiento en las mediciones, la mayoría de las estaciones no realizó la adecuada medición de la lluvia real en un número considerable de ocasiones. Por tal motivo, el valor de la medición en ese instante fue considerado como perdido.

La minería de datos será realizada utilizado el ambiente de aprendizaje automatizado Weka. Para ello primeramente se confeccionó una sencilla aplicación en Java la cual convierte los resultados de los modelos (los cuales eran numerosos y contenían las variables del modelo valorizadas para todos los horarios) a ficheros *.arff*, que son los que Weka puede manejar, ver ANEXO 8. Dicha aplicación es útil además pues filtra los datos eliminando todas aquellas instancias en las que, como se ha explicado, por algún motivo no se midió el valor de la lluvia real; y por tanto no es posible realizar una comparación con lo ocurrido realmente, ni tampoco asignarle una clase específica a una instancia en particular cuando se realiza su clasificación.

3.3.1 Clasificación de precipitaciones

En un primer encuentro con meteorólogos expertos, las clasificaciones iniciales de las precipitaciones que ellos sugirieron de acuerdo a la cantidad de milímetros caídos y a las características de nuestro país fueron las siguientes:

- ✓ DEBIL ($precipitaciones \leq 10mm$)
- ✓ MODERADA ($10mm < precipitaciones \leq 30mm$)
- ✓ MODERADA_INTENSA ($30mm < precipitaciones \leq 40mm$)
- ✓ INTENSA ($precipitaciones > 40mm$)

La base de casos conformada cuenta entonces con un total de 24 variables predictoras, y una variable objetivo que caracteriza el tipo de precipitación ocurrida clasificándola en cuatro clases. Los especialistas tienen un interés particular en pronosticar correctamente la clase INTENSA.

Dichas variables representan las salidas de los campos de: temperatura, altura geopotencial, humedad relativa en superficie y presión al nivel medio del mar, entre otros factores atmosféricos. Como se ha dicho, estas variables son todas numéricas, y son las salidas del modelo global GFS cada seis horas durante un plazo de 48 horas; además se incluyen como variables predictoras: el mes, el día y la lluvia pronosticada por dicho modelo general (la cual fue adicionada también a la base de casos de acuerdo al criterio experto).

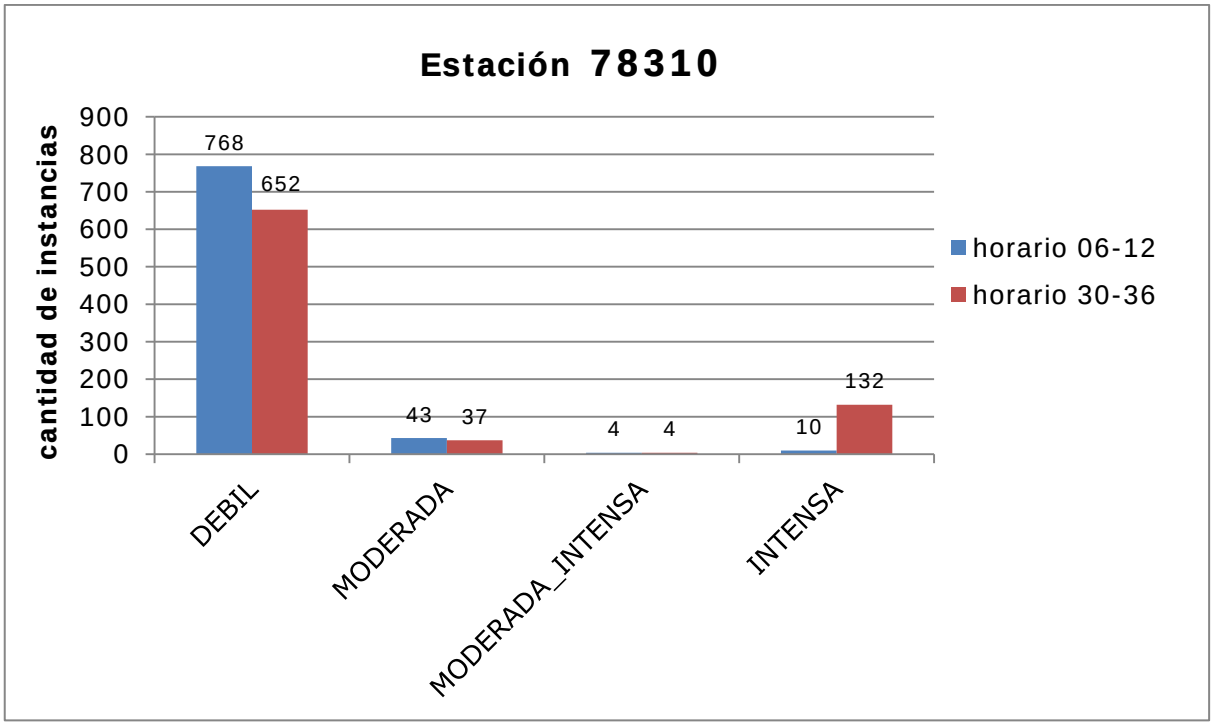
El período Lluvioso se consideró entre los meses de mayo y octubre, mientras el Poco Lluvioso entre los meses de noviembre y abril. Inicialmente la base de casos fue conformada con todos los días del año. No obstante, se consideraron especialmente los plazos de 06–12 horas y 30–36 horas, pues corresponden al período tarde–noche del

primer y segundo día respectivamente, horario durante el cual se producen estadísticamente el mayor número de eventos de precipitaciones.

El problema de pronóstico de precipitaciones fue clasificado primeramente usando algoritmos de aprendizaje automatizado tradicionales. Con motivo de dar una idea general de los resultados obtenidos, solamente se mostrarán los correspondientes a una sola de las 68 estaciones meteorológicas del país, aunque se obtuvieron resultados similares para el resto de las estaciones.

La Figura 3.3 presenta la cantidad de instancias por cada clase de la estación 78310 en los horarios comprendidos entre las 06–12 y las 30–36 horas de los años 2009, 2010, 2011 y 2012, los cuales corresponden al período tarde–noche (desde las 12*pm* hasta las 6*pm*) del primer y segundo día de pronóstico del modelo GFS. Se escogió este horario ya que es en el que estadísticamente más llueve durante el día, y por tanto presenta una mayor cantidad de instancias de las clases de interés (recordar que normalmente en el año el número de días que no llueve es mucho mayor que los que llueve).

Figura 3.3 Cantidad de instancias de cada clase para la estación 78310 en los horarios comprendidos entre 06–12 h y 30–36 h.



Como se puede apreciar, para esta estación en particular, durante el periodo 06–12 se tienen un total de 825 instancias, de las cuales 768 representan la clase de lluvia DEBIL,

43 de lluvia MODERADA, 4 de lluvia MODERADA_INTENSA y 10 de lluvia INTENSA. Durante el período 30–36 se tienen un total de 798 instancias, de las cuales 652 representan la clase de lluvia DEBIL, 37 de lluvia MODERADA, 4 de lluvia MODERADA_INTENSA y 132 de lluvia INTENSA. En el resto de las estaciones del país estos estadísticos presentan valores bastante similares a los de esta estación en particular.

En la primera experimentación realizada se utilizó validación cruzada con 10 particiones. Los estadísticos utilizados para las comparaciones entre algoritmos fueron: porciento de clasificación correcta, F-Measure y área ROC; en el caso de estos dos últimos se muestran sus resultados verticalmente de arriba hacia abajo representando a las clases DEBIL, MODERADA, MODERADA_INTENSA e INTENSA respectivamente, Tabla 3.1. Los algoritmos que mejores resultados arrojaron fueron el NaiveBayes, Multilayer Perceptron y kNN. El resto de los algoritmos, como por ejemplo los basados en árboles como el J48 y en máquinas vectoriales como el SMO arrojaron resultados mucho peores, por lo que no se incluyen en la tabla.

Tabla 3.1 Resultados de clasificación obtenidos usando los algoritmos tradicionales en la estación 78310, años 2009-2012.

Horario	Algoritmo	% de clasificación correcta	F-Measure	Área ROC
(06–12 horas) 12pm-6pm	**Naive Bayes**	73.9394	0.861	**0.783**
			0.162	0.622
			0.032	**0.833**
			0.108	0.668
	Multilayer Perceptron	**89.8182**	**0.947**	0.725
			0.081	**0.66**
			0	0.812
			0	**0.722**
	kNN	88	0.936	0.589
			0.225	0.598
			0	0.744
			0.087	0.516

Horario	Algoritmo	% de clasificación correcta	F-Measure	Área ROC
(30–36 horas) 12pm-6pm	**Naive Bayes**	59.2727	0.798	**0.791**
			0.106	0.581
			0.038	0.717
			0.16	0.672
	Multilayer Perceptron	**73.2121**	0.85	0.737
			0.067	0.708
			0	**0.939**
			0.262	**0.69**
	kNN	72.7273	**0.851**	0.621
			0.243	**0.581**
			0	0.748
			0.327	0.597

Como se puede apreciar en la tabla anterior, los resultados de los clasificadores tradicionales no fueron buenos. Aunque los porcientos de instancias bien clasificadas superaron el 73% en ambos horarios, los valores de los estadísticos F-Measure y Área ROC para las clases MODERADA, MODERADA_INTENSA e INTENSA (esta última, la de mayor interés) resultaron desfavorables, sobre todo en el caso del horario 06–12, en el cual se cuenta con una menor representación de estas clases en la base de casos.

Fusión de clases y reducción de instancias

Debido a que los resultados iniciales fueron desfavorables (como se muestra en la Tabla 3.1), se decidió en conjunto con los expertos del Instituto de Meteorología de Santa Clara flexibilizar el margen de lo que se considera lluvia INTENSA. Debido a que en el año una estación del país reporta pocas ocurrencias de eventos lluviosos, se hace difícil que los algoritmos de aprendizaje automático "aprendan" con tan pocas instancias de dicho evento meteorológico. Atendiendo a esta dificultad son redefinidas las clases de la siguiente forma:

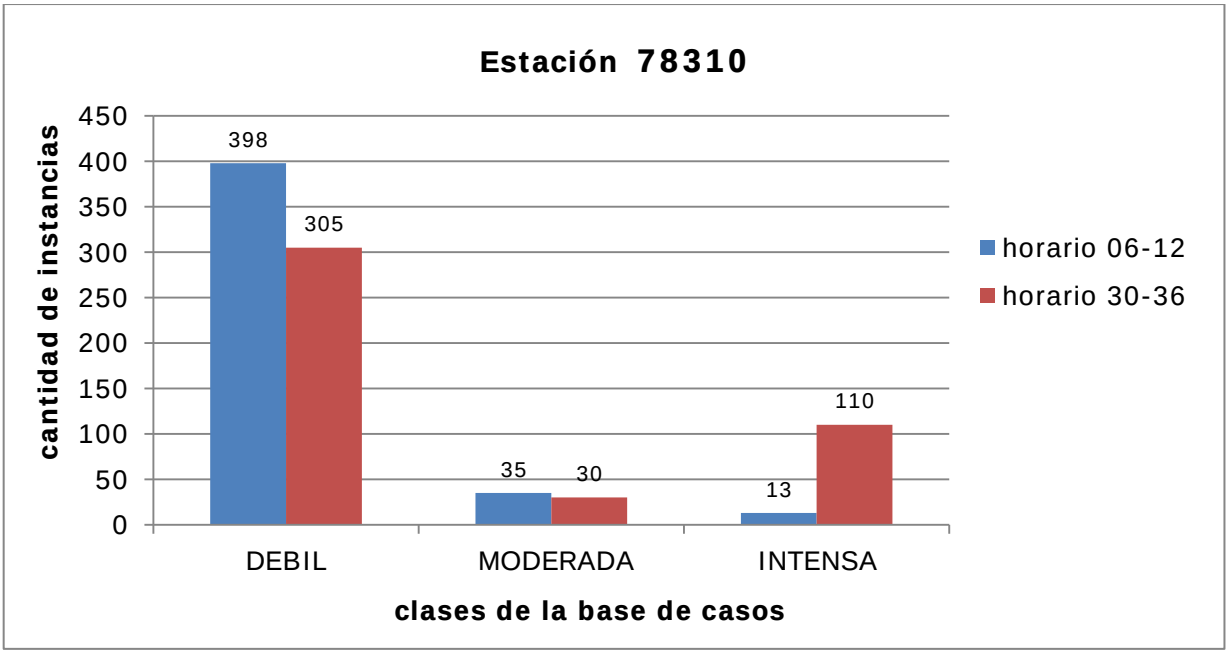
- ✓ DEBIL ($precipitaciones \leq 10mm$)
- ✓ MODERADA ($10mm < precipitaciones \leq 30mm$)
- ✓ INTENSA ($precipitaciones > 30mm$)

Consecuentemente se decidió realizar las siguientes modificaciones a la base de casos inicial:

- Fusionar las clases MODERADA_INTENSA e INTENSA en una sola clase denominada INTENSA.
- Considerar en los análisis solamente los meses lluviosos, comprendidos entre mayo y octubre de cada año.

La Figura 3.4 muestra cómo quedó conformada la base de casos después de haber sido realizados estos cambios. Como se puede apreciar, disminuyeron las ocurrencias de instancias clasificadas como DEBIL, mientras que aumentaron las clasificadas como MODERADA e INTENSA. Esto propició el aumento de representatividad de estas clases respecto a la base de casos inicial.

Figura 3.4 Cantidad de instancias de cada clase para la estación 78310 en los horarios comprendidos entre las 06–12 y las 30–36 horas después de haber fusionado y reducido instancias en la base de casos.



La Tabla 3.2 muestra los resultados obtenidos aplicando los algoritmos de aprendizaje automatizado tradicionales luego de haber fusionado y reducido instancias en la base de casos de la estación 78310, para equilibrar con ello la representatividad de sus clases. Como se puede apreciar, tanto los valores de F-Measure como de área ROC de las clases INTENSA y MODERADA mejoraron con respecto a la Tabla 3.1. Los mejores resultados los

muestra el algoritmo kNN para el segundo horario, pues clasificó mejor tanto las lluvias intensas como las moderadas.

Tabla 3.2 Resultados de clasificación obtenidos usando los algoritmos tradicionales después de haber fusionado y reducido instancias en la base de casos de la estación 78310 durante los años 2009-2012.

Horario	Algoritmo	% de clasificación correcta	F-Measure	Área ROC
(06–12 horas) 12pm-6pm	**Naive Bayes**	53.3632	0.706	**0.663**
			0.12	0.555
			0.093	**0.638**
	Multilayer Perceptron	82.7354	**0.906**	0.564
			0.065	0.487
			0	0.494
	kNN	80.0448	0.887	0.563
			0.225	**0.612**
			0	0.527
(30–36 horas) 12pm-6pm	**Naive Bayes**	45.3933	0.657	**0.658**
			0.142	**0.568**
			0.163	0.529
	Multilayer Perceptron	59.5506	0.739	0.595
			0.085	0.566
			0.271	0.563
	kNN	61.573	**0.741**	0.589
			0.2	0.549
			0.378	**0.604**

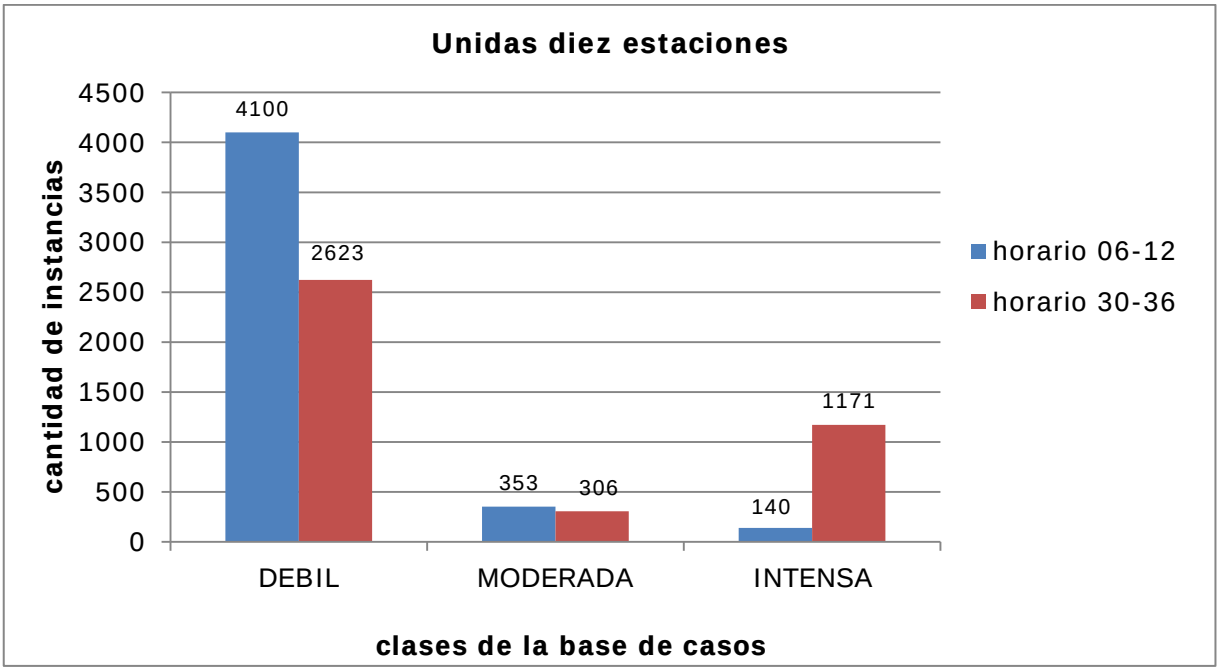
Fusión de varias estaciones

A pesar de que los resultados de clasificación mejoraron después de haber modificado la base de casos, Tabla 3.2 , estos aún distan mucho de ser confiables, sobre todo para la clase INTENSA que es la que más interesa en el pronóstico. No obstante, los dos experimentos que se han llevado a cabo hasta el momento sugieren que con el aumento

del número de instancias de la clase INTENSA se mejora la clasificación de las instancias de esa clase.

Partiendo de esta premisa y con el consentimiento de los expertos, se decidió fusionar en uno solo los horarios de 06–12 y 30–36 horas de 10 estaciones meteorológicas, con el objetivo de aumentar el número de instancias de la clase INTENSA. Se conformó entonces un conjunto de entrenamiento uniendo los datos de diez estaciones y un conjunto de control con los datos de otras diez estaciones diferentes, para los horarios 06–12 y 30–36. La Figura 3.5 muestra la cantidad de instancias de cada clase en los conjuntos de entrenamiento conformados.

Figura 3.5 Cantidad de instancias de cada clase después de unir los datos de diez estaciones en los horarios comprendidos entre las 06–12 h y 30–36 h.



La Tabla 3.3 muestra los resultados obtenidos al aplicar kNN (que es el algoritmo que mejores resultados arroja), utilizando el conjunto de entrenamiento conformado por diez estaciones meteorológicas para el aprendizaje, y por otras diez estaciones para el control.

Tabla 3.3 Resultados de clasificación obtenidos usando kNN después de unir los datos de diez estaciones en los horarios comprendidos entre 06–12 h y 30–36 h.

Horario	Algoritmo	% de clasificación correcta	F-Measure	Área ROC
(06–12 horas) 12pm-6pm	**kNN**	83.8293	0.915	0.535
			0.086	0.505
			0.106	0.536
(30–36 horas) 12pm-6pm	**kNN**	61.678	0.754	0.62
			0.117	0.535
			0.377	0.58

Reducción del desbalance entre las clases

Como se aprecia en la Figura 3.5, la cantidad de instancias clasificadas como DEBIL es mucho más numerosa que las clasificadas como MODERADA e INTENSA, a pesar de haber considerado de las estaciones los horarios más lluviosos del día. Debido a esto surge la necesidad de balancear la representatividad de estas dos últimas clases en el la base de casos. Para ello va a ser utilizado el filtro **NumerosityReduction** implementado en el Capítulo 2. Se redujo en un 20% el total de instancias del conjunto de control. Los resultados obtenidos se muestran en Tabla 3.4.

Tabla 3.4 Resultados de clasificación obtenidos usando kNN después de aplicar el filtro **NumerosityReduction** en el conjunto de entrenamiento.

Horario	Algoritmo	% de clasificación correcta	F-Measure	Área ROC
(06–12) 12pm-6pm	**kNN**	67.9512	0.81	0.698
			0.259	0.575
			0.245	0.561
(30–36) 12pm-6pm	**kNN**	53.2663	0.669	0.744
			0.109	0.524
			0.417	0.591

Como se puede apreciar, el uso del filtro **NumerosityReduction**, aunque disminuye los porcientos de instancias bien clasificadas, sin embargo aumenta ligeramente los valores

de los estadísticos F-Measure y área ROC de las clases MODERADA e INTESA. Siendo estas clases las que mayor interés reportan para la comunidad meteorológica en general.

3.3.2 Predicción de precipitaciones

El paquete **timeSeriesForecasting** de Weka posibilita realizar predicciones numéricas de un atributo seleccionado, a partir del aprendizaje de un conjunto de datos ordenado secuencialmente en el tiempo que lo contenga. Normalmente dicho aprendizaje se realiza mediante algoritmos que implementen algún tipo de regresión.

Se desea realizar con dicho paquete el pronóstico de precipitaciones a partir de los datos numéricos del modelo CBF, para ello se hace necesario primeramente cargar en Weka los datos correspondientes a una estación objetivo. De la estación cargada interesan especialmente los valores de precipitaciones reales ocurridas durante un periodo de tiempo dado; por lo tanto, es este el atributo cuyos valores futuros se van a pronosticar.

Para realizar la predicción con **timeSeriesForecasting** se ha seleccionado la estación 78310 en el horario comprendido entre las 30–36 horas. Como lo que se desea predecir son los valores de lluvia real para un período de tiempo dado, entonces la predicción debe aplicarse al atributo numérico LLUVIA. Dicha predicción se realiza como se muestra en el pseudocódigo de la Figura 3.6.

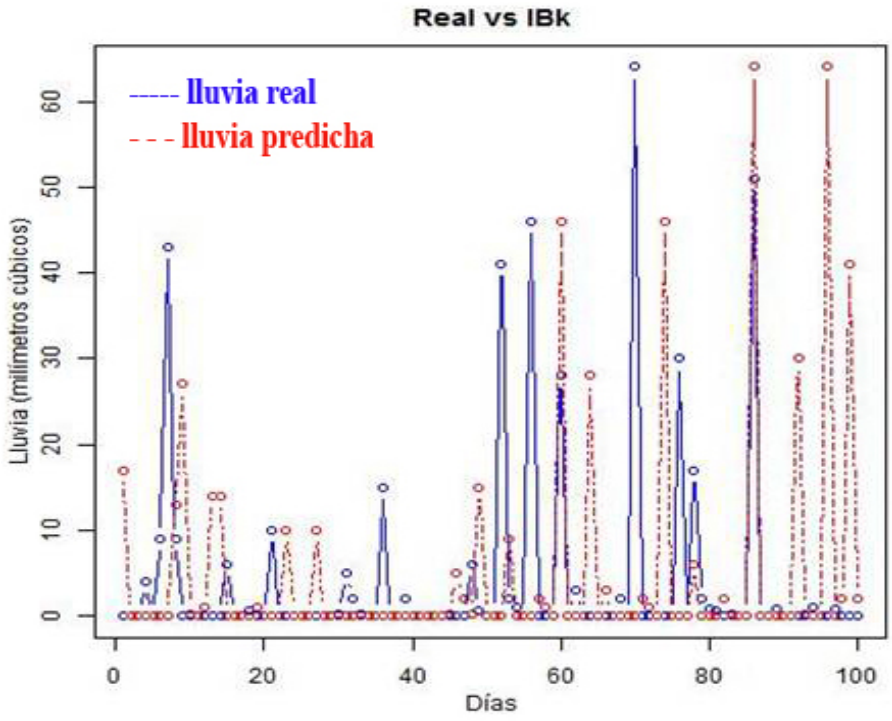
En la figura el entrenamiento se lleva a cabo omitiendo los valores de los últimos 100 días (líneas 3-5). Posteriormente, se usa un ciclo en el que se predicen los valores de lluvia para los dos días siguientes, actualizando en cada iteración la base de entrenamiento con los valores de precipitaciones reales correspondientes a los dos días predichos (líneas 8-15).

Figura 3.6 Algoritmo para la predicción de la lluvia usando el paquete **timeSeriesForecasting** de Weka.

```
PROCEDURE Predicción (List lluviaReal, List lluviaPronosticada)
    BEGIN
        totalDías <- LENGHT(lluviaReal)
        diaInicial <- totalDías - 100
        forecaster <- WekaForecaster (lluviaReal[1..diaInicial])
        i <- diaInicial + 1

        WHILE i < totalDías DO
        BEGIN

            lluviaPronosticada.add(forecaster.forecast(2))
            forecaster.update(lluviaReal[i..i+1])
            i <- i + 2

        END

    END
```

La Figura 3.7 muestra gráficamente los resultados obtenidos con el paquete utilizando el algoritmo kNN como base del aprendizaje automatizado. En el ANEXO 10 se muestran además las predicciones obtenidas utilizando otros algoritmos de aprendizaje.

Figura 3.7 Resultados obtenidos al aplicar el paquete **timeSeriesForecasting** de Weka usando el algoritmo de aprendizaje automatizado kNN.



Los estadísticos seleccionados para medir la calidad de la predicción fueron: Error Absoluto Relativo y Raíz del Error Cuadrático Medio. En la Tabla 3.5 se muestra la evaluación de la predicción utilizando cinco algoritmos de aprendizaje diferentes.

Tabla 3.5 Evaluación del error cometido en la predicción.

Algoritmo	Error absoluto relativo	Raíz del error cuadrático medio
Linnear Regression	1.07888165520421	12.0236807780393
SMOreg	**0.638723611790498**	12.207746439165
M5Rules	1.01018887107036	**11.5952492681725**
IBk	1.07982837994425	15.2675341820479
Multilayer Perceptron	1.40422948595443	16.3547462927163

Como se puede apreciar los resultados obtenidos con el modelo distan notablemente de los valores reales de precipitaciones acaecidas. De forma general, se aprecian altos errores en las predicciones obtenidas.

Estos experimentos permiten concluir que el modelo computacional definido mediante el uso del paquete **timeSeriesForecasting** de Weka no realiza un aprendizaje satisfactorio, y por tanto no consigue realizar una adecuada predicción de precipitaciones en la provincia de Villa Clara.

3.4 Conclusiones del capítulo

El pronóstico de precipitaciones a partir de modelos numéricos es un problema complejo. Los métodos computacionales utilizados actualmente por los institutos de meteorología para predecir los eventos de lluvias moderadas e intensas son muy imprecisos; debido a esto se requiere en gran medida del criterio experto a la hora de emitir un pronóstico fiable.

Se abordó el problema del pronóstico de precipitaciones, a partir de los datos de salida generados por el modelo numérico global GFS, con el uso de técnicas de aprendizaje automatizado. El problema se modeló de dos formas: utilizando un esquema de clasificación tradicional y utilizando un esquema de regresión para series temporales. El ambiente de aprendizaje automatizado utilizado fue el Weka. Sobre la base del análisis de los resultados obtenidos, se arribó a las siguientes conclusiones:

- ✓ Con el paquete **timeSeriesForecasting** de Weka no se consigue una predicción confiable de las precipitaciones.
- ✓ El desbalance presente entre las distintas clases que caracterizan la cantidad de precipitaciones, en especial de la clase INTENSA, afecta sensiblemente el aprendizaje de los métodos utilizados.
- ✓ Una vez mitigado el efecto del desbalance existente, solo se consiguen mejoras leves de los clasificadores.
- ✓ Las técnicas de aprendizaje automatizado utilizadas en este Capítulo no mejoran los resultados de los modelos numéricos que se utilizan actualmente en el Instituto de Meteorología de Santa Clara.

CONCLUSIONES DEL TRABAJO

A partir la investigación llevada a cabo, y de los resultados obtenidos en este trabajo se arriba a las siguientes conclusiones:

- ✓ Las técnicas de minería de datos para series temporales superan muchas de las limitaciones de los métodos estadísticos tradicionales.
- ✓ Atendiendo a las deficiencias de Weka para efectuar clasificación de series temporales, se implementó en forma de paquete una selección de métodos para facilitar esta tarea.
- ✓ Las técnicas de aprendizaje automatizado aplicadas al pronóstico de precipitaciones no mejoran los resultados de los modelos numéricos que se utilizan actualmente en el Instituto de Meteorología de Santa Clara.

RECOMENDACIONES

A partir del estudio realizado, y derivadas de las conclusiones de este trabajo, se plantean las siguientes recomiendaciones:

- ✓ Extender el paquete **timeSeriesClassification** con herramientas que posibiliten realizar otras tareas de minería de datos para series temporales.
- ✓ Mejorar la función de distancia **DTWDistance** con un algoritmo que sea capaz de determinar el ancho óptimo de la banda Sakoe-Chiba.
- ✓ Aplicar otras técnicas a la base de casos de las salidas del modelo GFS que permitan manejar el desbalance entre sus clases.

REFERENCIAS BIBLIOGRÁFICAS

1. Fu, T.-c., *A review on time series data mining.* Engineering Applications of Artificial Intelligence, 2011. **24**(1): p. 164-181.
2. Shahiduzzaman, M. and K. Alam, *Cointegration and causal relationships between energy consumption and output: Assessing the evidence from Australia.* Energy Economics, 2012.
3. Pankratz, A., *Forecasting with dynamic regression models*. 1991: Wiley Online Library.
4. Pampu, N.C., *STUDY OF EFFECTS OF THE SHORT TIME FOURIER TRANSFORM CONFIGURATION ON EEG SPECTRAL ESTIMATES.* power. **4**: p. 6.
5. Lee, Y.W., T.P. Cheatham Jr, and J.B. Wiesner, *Application of correlation analysis to the detection of periodic signals in noise.* Proceedings of the IRE, 1950. **141**(10): p. 1165-1171.
6. Pandit, S.M. and S.-M. Wu, *Time series and system analysis with applications*. 1990: RE Krieger Publishing Company.
7. Bowerman, B.L. and R.T. O'Connell, *Forecasting and time series: an applied approach*. 1993: Belmont.
8. Ljung, G.M. and G.E.P. Box, *On a measure of lack of fit in time series models.* Biometrika, 1978. **65**(2): p. 297-303.
9. Box, G.E.P. and G.M. Jenkins, *Time series analysis: Forecasting and control (rev. ed.) Holden-Day.* San Francisco, 1976. **575**.
10. Havenner, A. and P. Swamy, *A random coefficient approach to seasonal adjustment of economic time series.* Journal of Econometrics, 1981. **15**(2): p. 177-209.
11. Larose, D.T., *Discovering knowledge in data: an introduction to data mining*. 2004: Wiley-Interscience.
12. Bigus, J.P., *Data mining with neural networks: solving business problems from application development to decision support*. 1996: McGraw-Hill, Inc.
13. Han, J. and M. Kamber, *Data mining: concepts and techniques*. 2006: Morgan Kaufmann.
14. *Aussie Gold History*. 1998; Available from: http://www.uq.net.au/~zzdvande/history.html.
15. Gabel, R.A. and R.A. Roberts, *Signals and linear systems*. 2009: John Wiley & Sons.
16. Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, *From data mining to knowledge discovery in databases.* AI magazine, 1996. **17**(3): p. 37.
17. Weiss, S.M. and N. Indurkhya, *Predictive data mining: a practical guide*. 1998: Morgan Kaufmann.
18. Abarbanel, H.D.I. and J.P. Gollub, *Analysis of observed chaotic data.* Physics Today, 1996. **49**: p. 86.
19. Iwanski, J.S. and E. Bradley, *Recurrence plots of experimental data: To embed or not to embed?* Chaos: An Interdisciplinary Journal of Nonlinear Science, 1998. **8**(4): p. 861-871.

20. Povinelli, R.J. *Using genetic algorithms to find temporal patterns indicative of time series events*. 2000.
21. Faloutsos, C., M. Ranganathan, and Y. Manolopoulos, *Fast subsequence matching in time-series databases*. Vol. 23. 1994: ACM.
22. Keogh, E. *Fast similarity search in the presence of longitudinal scaling in time series databases*. 1997. IEEE.
23. Keogh, E.J. and M.J. Pazzani, *A simple dimensionality reduction technique for fast similarity search in large time series databases*, in *Knowledge Discovery and Data Mining. Current Issues and New Applications*. 2000, Springer. p. 122-133
24. Abonyi, J., et al. *Principal component analysis based time series segmentation-Application to hierarchical clustering for multivariate process data*. 2003.
25. Åström, K.J., *On the choice of sampling rates in parametric identification of time series.* information Sciences, 1969. **1**(3): p. 273-278.
26. Yi, B. *K., & Faloutsos, C.(2000). Fast time sequence indexing for arbitrary Lp norms*.
27. Keogh, E., et al. *Locally adaptive dimensionality reduction for indexing large time series databases*. 2001. ACM.
28. Keogh, E.J. and M.J. Pazzani. *Derivative dynamic time warping*. 2001.
29. Smyth, P. and E. Keogh. *Clustering and mode classification of engineering time series data*. 1997. Citeseer.
32. Geurts, P., *Pattern extraction for time series classification*, in *Principles of Data Mining and Knowledge Discovery*. 2001, Springer. p. 115-127.
33. Zhang, H., T.B. Ho, and M.S. Lin, *A non-parametric wavelet feature extractor for time series classification*, in *Advances in knowledge discovery and data mining*. 2004, Springer. p. 595-603.
34. Xi, X., et al. *Fast time series classification using numerosity reduction*. 2006. ACM.
35. Povinelli, R.J., et al., *Time series classification using Gaussian mixture models of reconstructed phase spaces.* Knowledge and Data Engineering, IEEE Transactions on, 2004. **16**(6): p. 779-783.
36. Rodríguez, J.J. and C.J. Alonso. *Interval and dynamic time warping-based decision trees*. 2004. ACM.
37. Wei, L., E. Keogh, and X. Xi. *SAXually explicit images: Finding unusual shapes*. 2006. IEEE.
38. Agrawal, R., et al., *System and method for discovering similar time sequences in databases*, 1999, Google Patents.
39. Megalooikonomou, V., et al. *A multiresolution symbolic representation of time series*. 2005. IEEE.
40. Bernad, D.J., *Finding patterns in time series: a dynamic programming approach.* Advances in knowledge discovery and data mining, 1996.
41. Kruskall, J.B., Liberman, *The symmetric time warping algorithm: from continuous to discrete,TimeWarps,String Edits and Macromolecules.* 1983.
42. Ratanamahatana, C.A. and E. Keogh. *Making time-series classification more accurate using learned constraints*. 2004. Lake Buena Vista, Florida.

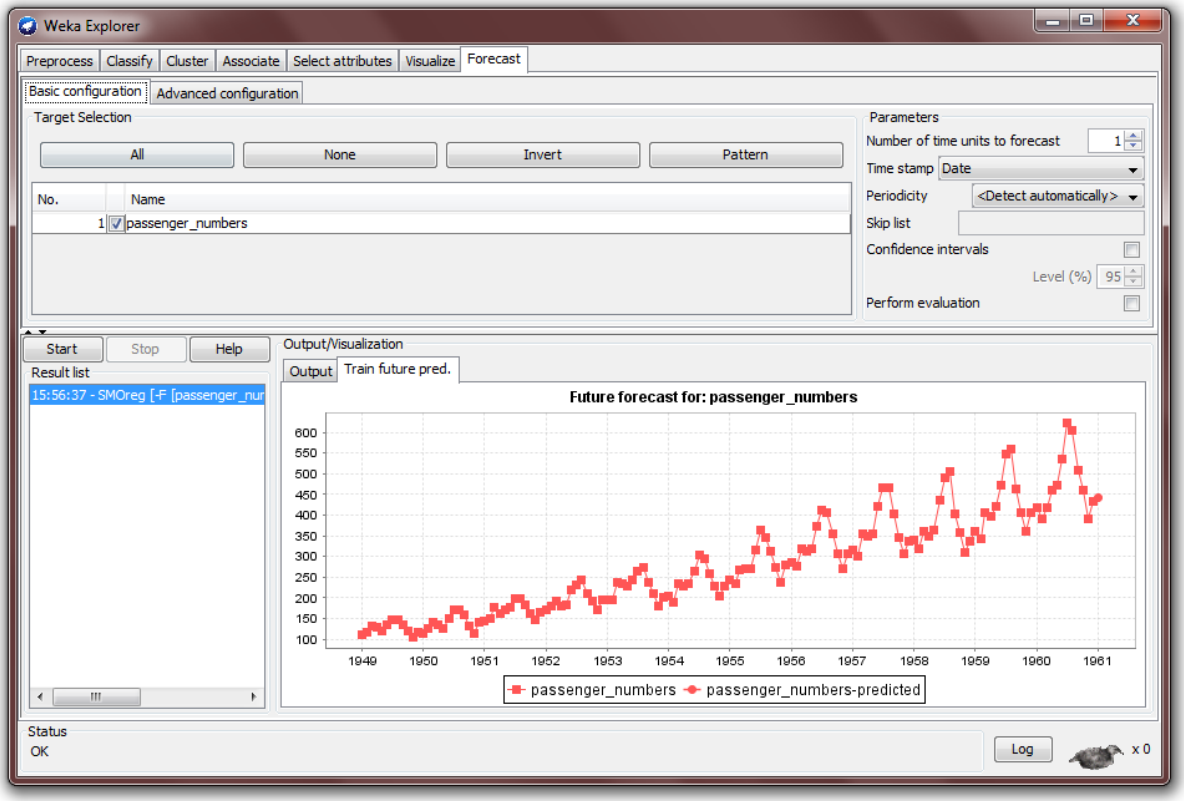
43. Sakoe, H. and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 1978. **26**(1): p. 43-49.
44. Keogh, E., S. Lonardi, and B.Y.-c. Chiu. *Finding surprising patterns in a time series database in linear time and space*. 2002. ACM.
45. Ding, H., et al., *Querying and mining of time series data: experimental comparison of representations and distance measures.* Proceedings of the VLDB Endowment, 2008. **1**(2): p. 1542-1552.
46. Agrawal, R., C. Faloutsos, and A. Swami, *Efficient similarity search in sequence databases*. 1993: Springer.
47. Moon, Y.-S., K.-Y. Whang, and W.-K. Loh. *Duality-based subsequence matching in time-series databases*. 2001. IEEE.
48. Moon, Y.-S., K.-Y. Whang, and W.-S. Han. *General match: a subsequence matching method in time-series databases based on generalized windows*. 2002. ACM.
49. Han, W.-S., et al. *Ranked subsequence matching in time-series databases*. 2007. VLDB Endowment.
50. Das, G., et al., *Rule discovery from time series.* Knowledge Discovery and Data Mining, 1998: p. 16-22.
52. Hochheiser, H. and B. Shneiderman, *Dynamic query tools for time series data sets: timebox widgets for interactive exploration.* Information Visualization, 2004. **3**(1): p. 1-18
53. Keogh, E., H. Hochheiser, and B. Shneiderman, *An augmented visual query mechanism for finding patterns in time series data*, in *Flexible Query Answering Systems*. 2002, Springer. p. 240-250.
54. Van Wijk, J.J. and E.R. Van Selow. *Cluster and calendar based visualization of time series data*. 1999. IEEE.
55. Weber, M., M. Alexa, and W. Müller. *Visualizing time-series on spirals*. 2001.
56. Caraça-Valente, J.P. and I. López-Chavarrías. *Discovering similar patterns in time series*. 2000. ACM.
57. Lerner, A., et al. *Fast algorithms for time series with applications to finance, physics, music, biology, and other suspects*. 2004. ACM.
58. Ma, J. and S. Perkins. *Online novelty detection on temporal sequences*. 2003. ACM.
59. Chan, P.K. and M.V. Mahoney. *Modeling multiple time series for anomaly detection*. 2005. IEEE.
60. Oates, T. *Identifying distinctive subsequences in multivariate time series by clustering*. 1999. ACM.
61. Wang, H., et al. *Clustering by pattern similarity in large data sets*. 2002. ACM.
62. Panuccio, A., M. Bicego, and V. Murino, *A Hidden Markov Model-based approach to sequential data clustering*, in *Structural, Syntactic, and Statistical Pattern Recognition*. 2002, Springer. p. 734-743.
63. Lagus, K., et al. *Self-organizing maps of document collections: A new approach to interactive exploration*. 1996. Menlo Park, CA: AAAI.

64. Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. & Ratanamahatana, C. A. *The UCR Time Series Classification/Clustering Homepage*. 2011; Available from: www.cs.ucr.edu/~eamonn/time_series_data/
65. Ratanamahatana, C.A. and E. Keogh. *Everything you know about dynamic time warping is wrong*. 2004.
66. Rath, T.M. and R. Manmatha. *Word image matching using dynamic time warping*. 2003. IEEE.
67. Witten, I.H., E. Frank, and M.A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. 2011: Morgan Kaufmann.
68. Group, M.L. *Weka 3: Data Mining Software in Java Machine Learning Group at University of Waikato.* 2013 [cited 2013 May 15th]; Available from: http://www.cs.waikato.ac.nz/ml/weka/.
69. COMMUNITY, P. *Time Series Analysis and Forecasting with Weka*. 2013 [cited 2013 May 23]; Available from: http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka.
70. Foundation, A.S. 2013 [cited 2013 June 26]; Available from: http://ant.apache.org.
71. Héctor Matías Gonzále, L.I.A.P., *Extensiones al Ambiente de Aprendizaje Automatizado WEKA*, in *Departamento de Inteligencia Artificial*2005-2006, Universidad Central Marta Abreu de las Villas: Santa Clara.
72. Ye, L. and E. Keogh. *Time series shapelets: a new primitive for data mining*. 2009. ACM.
73. Kadous, M.W. *Learning comprehensible descriptions of multivariate time series*. 1999.
74. Keogh, E.J. and M.J. Pazzani. *Scaling up dynamic time warping for datamining applications*. 2000. ACM.
75. Yi, B.-K., H.V. Jagadish, and C. Faloutsos. *Efficient retrieval of similar time sequences under time warping*. 1998. IEEE.
76. Jiang, L., et al. *Survey of improving k-nearest-neighbor for classification*. 2007. IEEE.
77. Rodríguez, J.J., C.J. Alonso, and H. Boström, *Learning first order logic time series classifiers: Rules and boosting*, in *Principles of Data Mining and Knowledge Discovery*. 2000, Springer. p. 299-308.
78. Nanopoulos, A., R. Alcock, and Y. Manolopoulos, *Feature-based classification of time-series data.* Information processing and technology, 2001: p. 49-61.
79. Ratanamahatana, C.A. and E. Keogh. *Three myths about dynamic time warping data mining*. 2005.
80. Pękalska, E., R.P.W. Duin, and P. Paclík, *Prototype selection for dissimilarity-based classifiers.* Pattern Recognition, 2006. **39**(2): p. 189-208.
81. Wilson, D.R. and T.R. Martinez. *Instance pruning techniques*. 1997. MORGAN KAUFMANN PUBLISHERS, INC.
82. Demšar, J., *Statistical comparisons of classifiers over multiple data sets.* The Journal of Machine Learning Research, 2006. **7**: p. 1-30.

ANEXOS

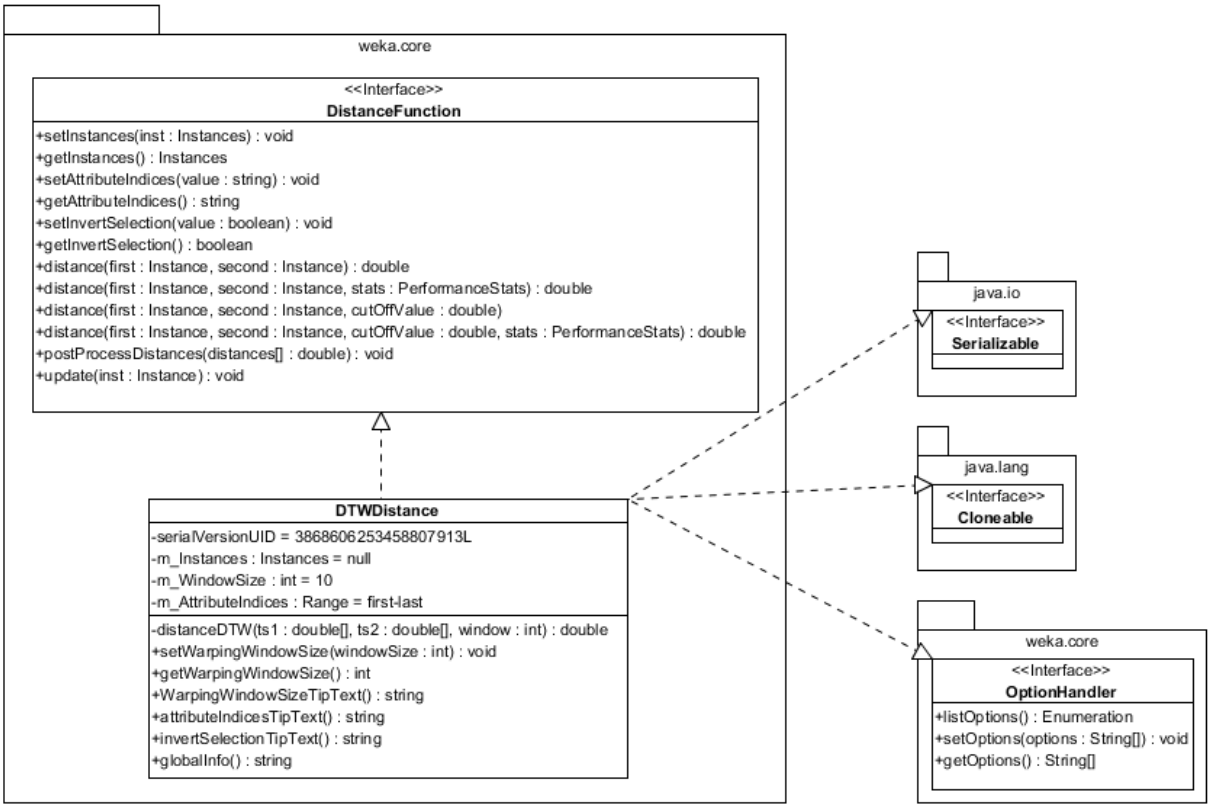
ANEXO 1

Interfaz gráfica del paquete **timeSeriesForecasting** de Weka.



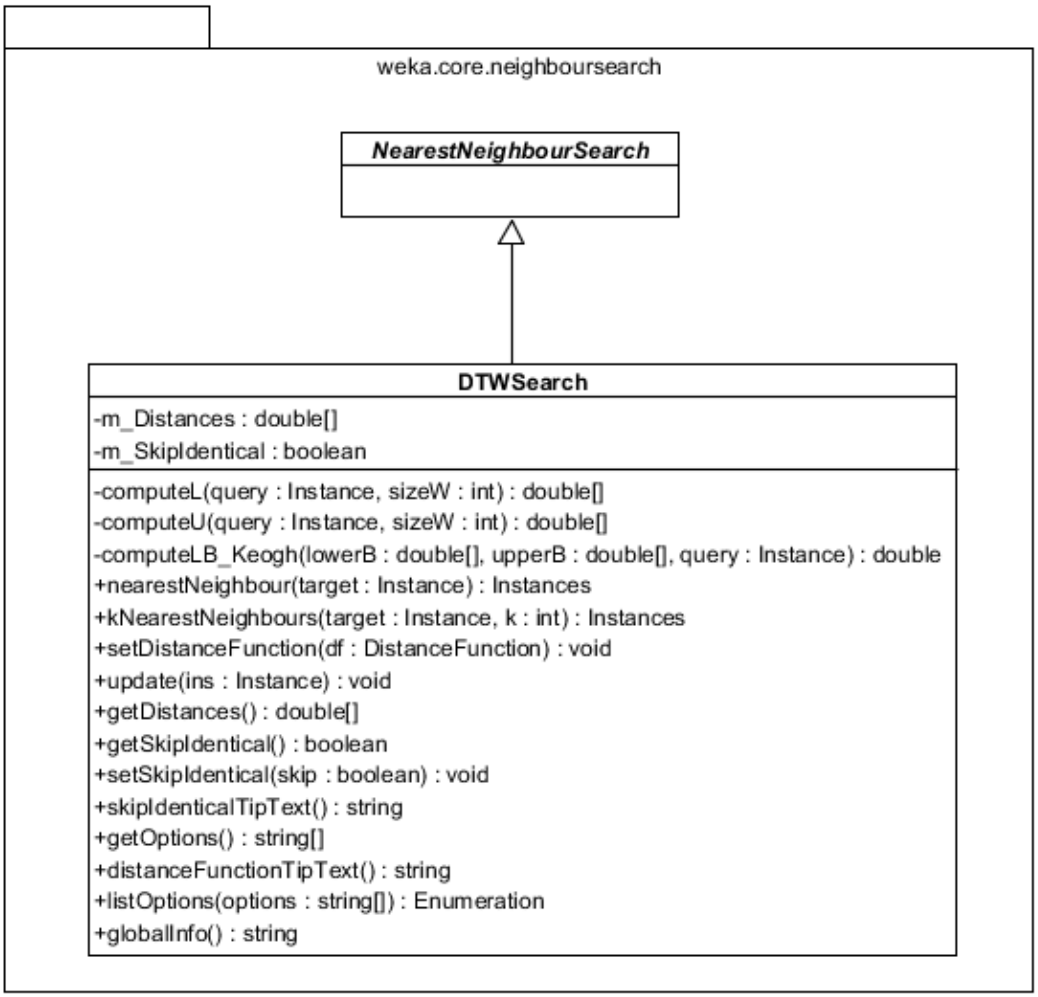
ANEXO 2

Diagrama de clases de la función de distancia **DTWDistance** implementada en Weka.



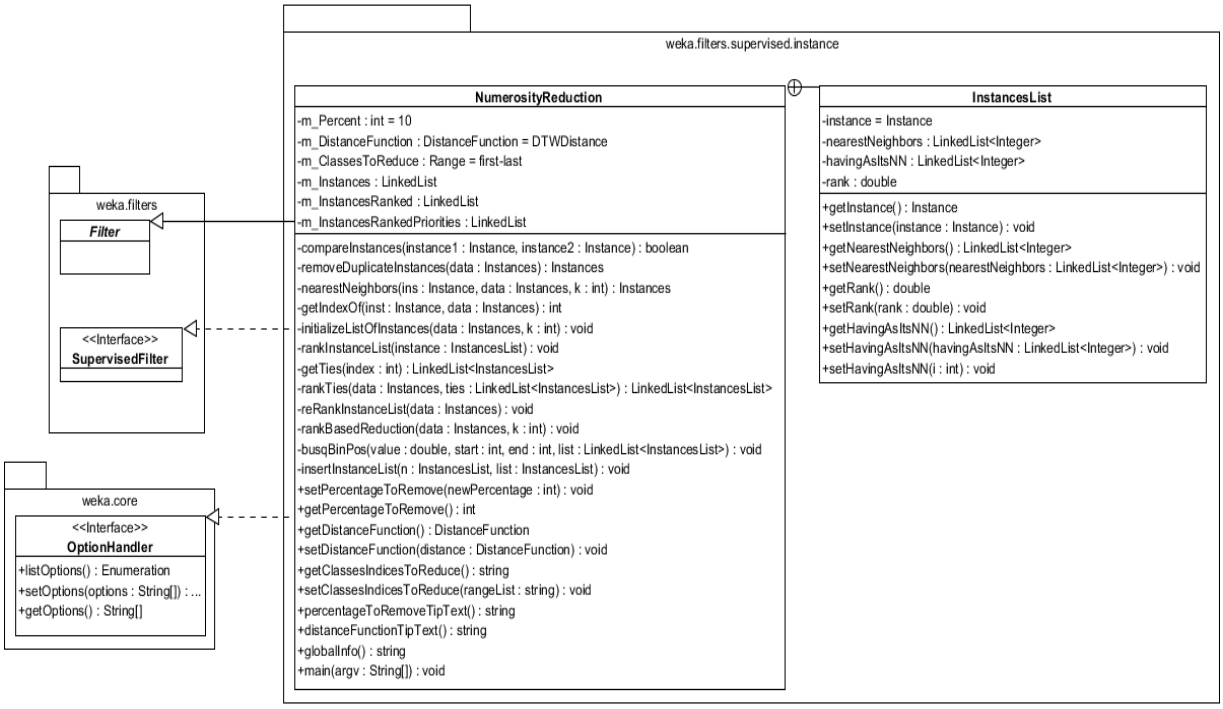
ANEXO 3

Diagrama de clases del algoritmo de búsqueda de *k* vecinos más cercanos **DTWSearch** implementado en Weka.



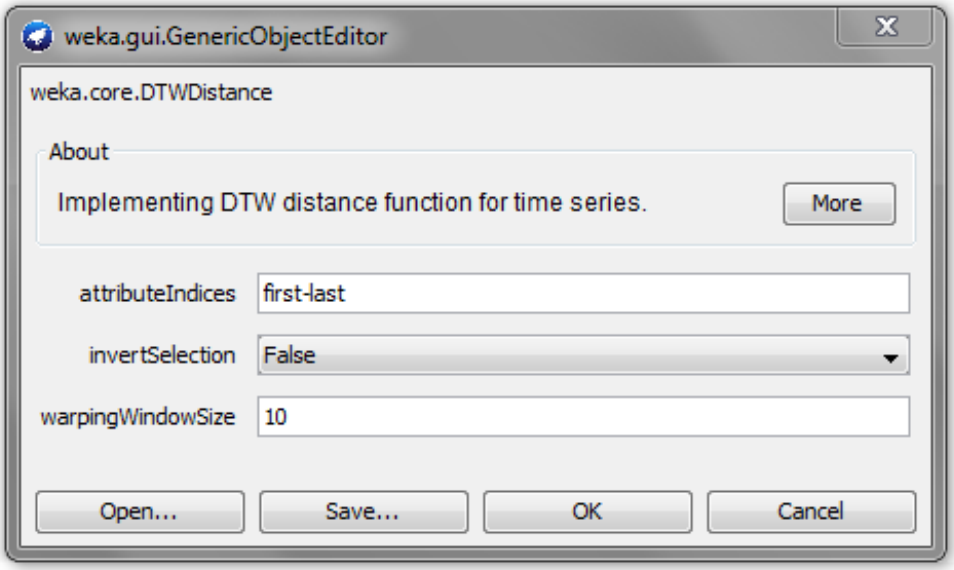
ANEXO 4

Diagrama de clases del filtro **NumerosityReduction** implementado en Weka.



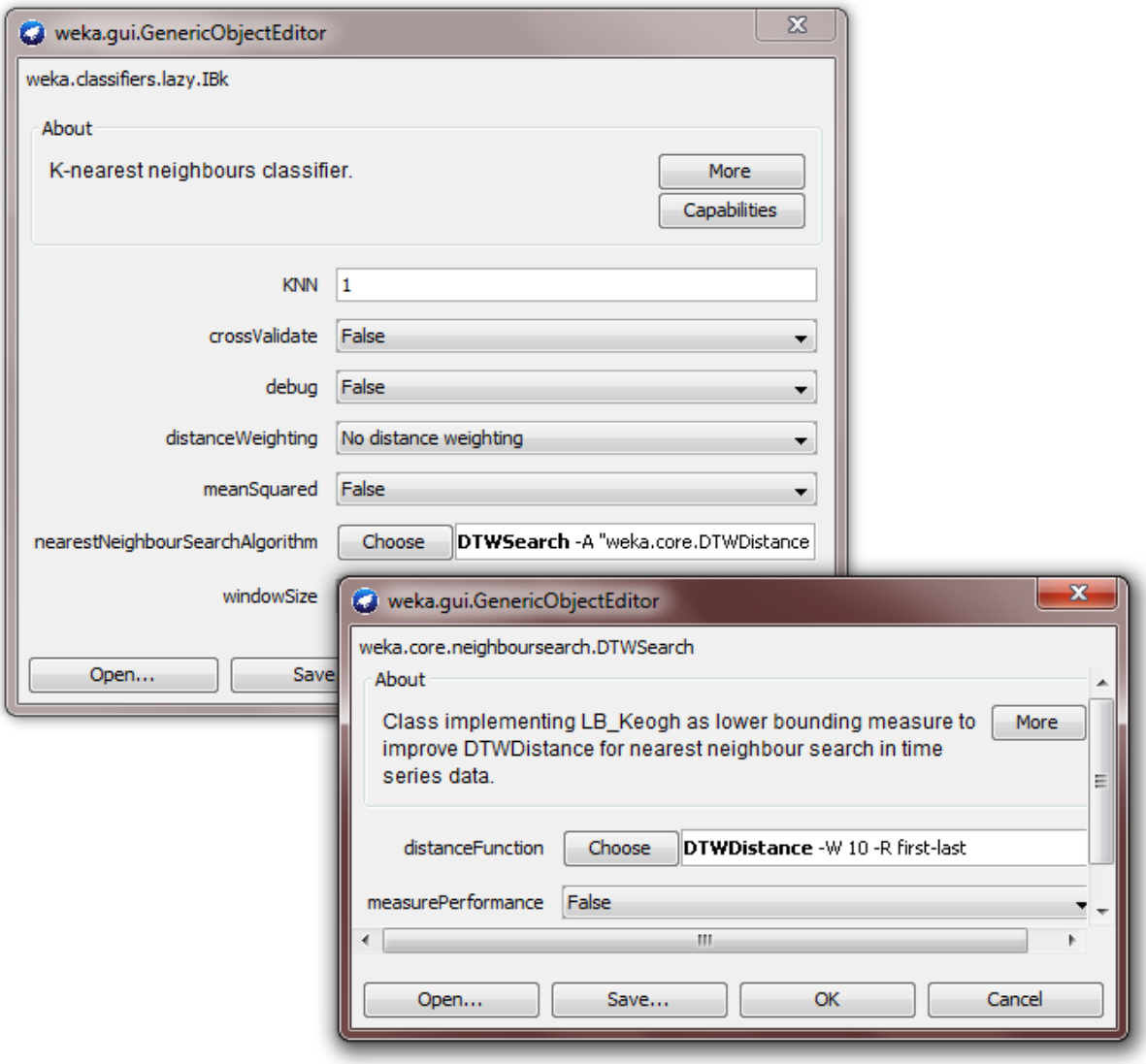
ANEXO 5

Interfaz gráfica de la función de distancia **DTWDistance** implementada en Weka.



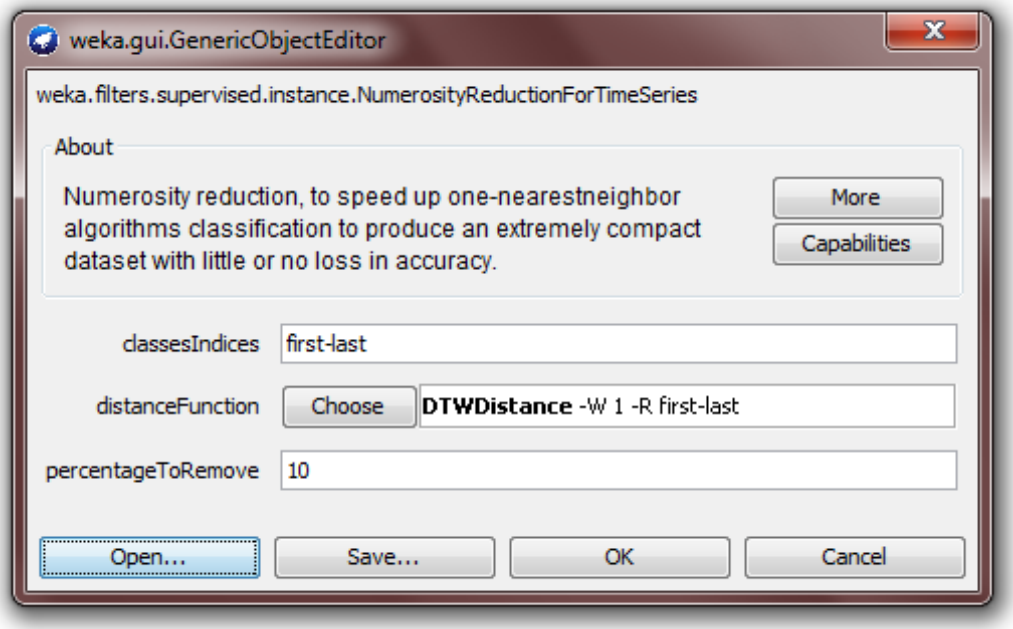
ANEXO 6

Interfaz gráfica del algoritmo de búsqueda de vecinos más cercanos **DTWSearch** implementado en Weka.



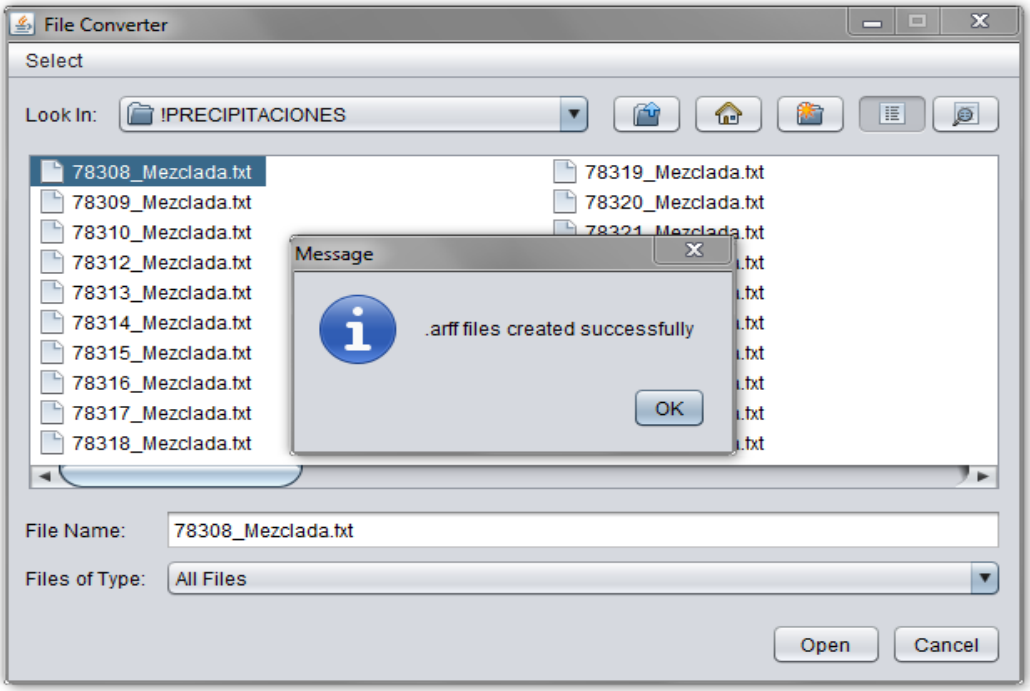
ANEXO 7

Interfaz gráfica del filtro para la reducción de la numerosidad **NumerosityReduction** implementado en Weka.



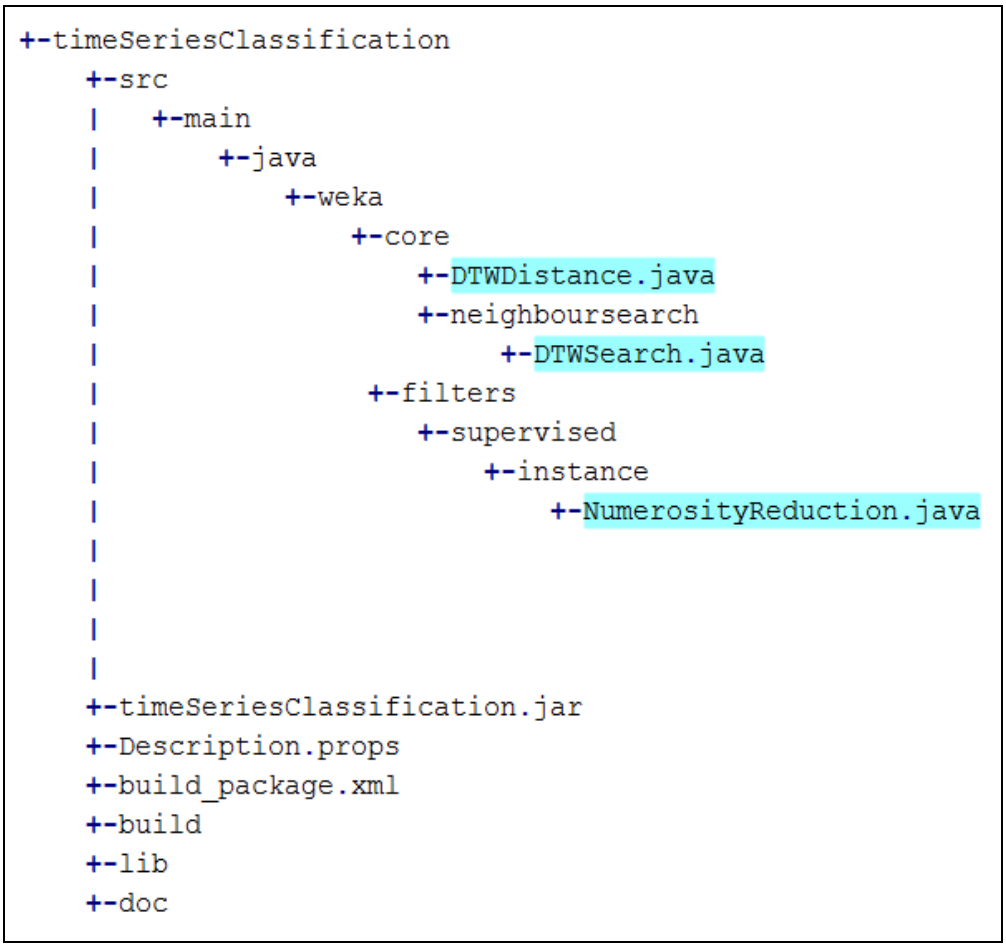
ANEXO 8

Aplicación que convierte los datos de salida del modelo GBF a formato *.arff*.



ANEXO 9

Estructura del paquete **timeSeriesClassification**



ANEXO 10

Resultados obtenidos usando el paquete **timeSeriesForecasting** con diferentes algoritmos de aprendizaje.