k-Vecinos más Cercanos Difuso para Clasificación Monotónica

Sergio González, Salvador García, Francisco Herrera
Dept. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada
Granada, España
{sergiogvz, salvagl, herrera}@decsai.ugr.es

Sheng-Tun Li
Dept. de Gestión Industrial y de Información
Universidad Nacional Cheng Kung
Tainan, Taiwan
stli@mail.ncku.edu.tw

Robert John
Escuela de Ciencias de la Computación
Universidad de Nottingham
Nottingham, Reino Unido
robert.john@nottingham.ac.uk

***Resumen*—La clasificación con restricciones monotónicas proviene de una necesidad de algunos problemas reales, en los que la variable de salida no disminuyen con un aumento de las entrada, o viceversa. Los conjuntos de datos reales con frecuencia tiene una gran cantidad de ruido de clase. Este incrementa el número de violaciones monotónicas y repercute negativamente en el rendimiento de los clasificadores. Por lo que, algunos métodos deben recurrir a técnicas de re-etiquetado que pueden alterar el conocimiento del problema. Además, la inclusión de estas restricciones obligan a los clasificadores a diversificar sus objetivos entre precisión y monotonicidad.**

Con esta contribución, se propone un nuevo modelo basado en los k-Vecinos más Ceranos Difuso para la clasificación con restricciones monotónicas, MonFkNN. Este clasificador incorpora un nuevo cálculo de pertenencias difusas, que lo dota de robustez frente ruido monotónico sin necesidad de re-etiquetado. Asimismo, se ha diseñado para ser adaptable a las diferentes necesidades del problema. Tras varios estudios experimentales, ha demostrado ser tremendamente más preciso igualando el grado de monotonicidad de los mejores métodos de la literatura.

***Palabras Clave*—Ciencia de datos, aprendizaje perezoso, vecinos más cercanos difuso, restricciones monotónicas, clasificación ordinal, regresión ordinal**

I. Introducción

La evaluación de activos o individuos [1], [2] tienen como objetivo determinar los artículos más valiosos según sus virtudes, es decir, la clasificación en etiquetas ordinales según sus atributos ordinales. Además, estas aplicaciones suelen conllevar una restricción monotónica entre las entradas y la clase. Es decir, la predicción de clase de un individuo no debe disminuir con un mejor valor para una determinada variable, fijando el resto. Estos problemas se conocen como clasificación con restricciones de monotonicidad [3] y la ruptura de estas, como violaciones de monotonicidad.

Este trabajo ha contado con el apoyo del Proyecto Español TIN2017-89517-P y una beca de investigación (FPU) otorgada a S. González por el MECD.

Estas tareas de aprendizaje exigen otros requisitos además de modelos precisos, como la consistencia monótona de las predicciones o la minimización de los errores de clasificación. Sin embargo, estos otros objetivos pueden perjudicar la precisión. Por lo tanto, se debe buscar un equilibro justo entre las diferentes necesidades de cada problema.

En estos últimos años, se ha diseñado nuevos algoritmos que consideran estas restricciones [1], [3]–[6]. El aprendizaje basado en la instancia ha demostrado ser una buena aproximación para la clasificación monotónica [4], [6]. Sin embargo, algunos de estos métodos, como la variante monotónica de k-Vecinos más Cercanos [4] (MkNN), necesitan aprender de un conjunto completamente monótono para asegurar predicciones monótonas. Esta circunstancia rara vez se cumple en conjuntos reales, donde el ruido y las discrepancias son comunes. Por ello, se hace uso de estrategias de re-etiquetado que alteran las etiquetas de clase de algunos ejemplos forzando un conjunto monotónico [4].

En clasificación estándar, la versión difusa de los k-Vecinos más Cercanos [7] (FkNN) es un método muy sólido con un gran rendimiento, gracias a su alta robustez al ruido [8]. En esta contribución, se presenta un nuevo modelo diseñado sobre FkNN con nociones de MkNN para tener en cuenta las restricciones de monotonicidad, llamado MonFkNN. La robustez de FkNN se ha adaptado para reducir el impacto de las violaciones monotónicas sin la necesidad de re-etiquetado. Adicionalmente, se han añadido otras técnicas para tratar las pertenencias de instancia de una manera más adecuada según la monotonicidad. Nuestra propuesta MonFkNN es un clasificador flexible que cubre diferentes necesidades de monotonicidad y precisión, con buen equilibrio entre ambos, ajustando sus parámetros.

Este documento está organizado de la siguiente manera. En la Sección II, presentamos el problema de la clasificación con restricciones monotónicas y los métodos relacionados con

nuestra propuesta. La sección III está dedicada a explicar con lujo detalle la propuesta MonFkNN. En la Sección IV, se presenta el marco experimental utilizado en los diferentes estudios empíricos. La sección V plantea tres estudios experimentales: un análisis sobre los parámetros de nuestro método y sus dos configuraciones más relevantes, una profunda comparación con el Estado-del-Arte y un estudio de la robustez de MonFkNN frente al ruido. Finalmente, en la Sección VI, se presentan las principales conclusiones de este estudio.

II. Preliminares

En esta sección, se introducen los preliminares sobre los que se construye este trabajo.

II-A. Clasificación Monotónica

La clasificación monotónica tiene como objetivo predecir la etiqueta de la clase y a partir del vector de la característica de entrada x, donde $y \in \mathcal{Y} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_c\}$, $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$. Como propiedad principal, los atributos y sus predicciones están monotónicamente restringidos por conocimiento previo del problema, es decir, $\mathbf{x} \succeq \mathbf{x}' \rightarrow f(\mathbf{x}) \geq f(\mathbf{x}')$ [9], donde $\mathbf{x} \succeq \mathbf{x}'$ implica $\forall_{j=1,\ldots,K}, \mathbf{x}_j \geq \mathbf{x}'_j$, esto es, $\mathbf{x}$ domina $\mathbf{x}'$. Por lo tanto, el objetivo principal es construir clasificadores que no violen estas restricciones, conocidos como clasificadores monótonos.

Se pueden distinguir dos tipos diferentes de clasificadores monotónicos: los modelos monotónicos aproximados, que minimizan el número de violaciones monotónicas en sus decisiones, y los clasificadores monotónicos puros, cuyas predicciones son siempre monotónicas en relación con el entrenamiento. Esto último es difícil de lograr, especialmente en aplicaciones de la vida real, donde los conjuntos de datos de formación rara vez son puramente monótonos. Para ser considerado monotónico, un conjunto de datos debe tener todos sus pares de instancias monotónicas entre sí [6]: $\mathbf{x}_i \succeq \mathbf{x}_j \rightarrow y_i \geq y_j, \forall_{i,j}$.

II-B. k-Vecinos más Cercanos Monotónico

La aproximación de los k-Vecinos más Cercanos para la clasificación monotónica [4] (MkNN) modifica la regla estándar del vecino más cercano para evitar violaciones monotónicas en sus predicciones. Para ello, MkNN calcula para cada nuevo ejemplo x_i el rango $r_i = [y_{min}, y_{max}]$ de etiquetas de clase válidas, que satisfagan las restricciones monotónicas. El y_{min} inferior de r_i se calcula como la etiqueta de clase más alta de todas las instancias por debajo del ejemplo x_i. Análogamente, el límite superior y_{max} es la etiqueta de clase mínima de las instancias superiores a x_i (véase la Ec. 1).

$$r_i = \begin{cases} y_{min} = \text{máx}\{y | (x,y) \in \mathcal{D} \wedge x_i \succeq x\} \\ y_{max} = \text{mín}\{y | (x,y) \in \mathcal{D} \wedge x \succeq x_i\} \end{cases} \quad (1)$$

Se pueden distinguir dos variantes diferentes dependiendo de cómo se extraen los vecinos para una nueva instancia i. La variante *enRango* considera los ejemplos k más cercanos x_j con sus etiquetas de clase y_j en el rango $[y_{min}, y_{max}]$. La versión *fueraDeRango* extrae los vecinos, independientemente de su clases, y luego, se eliminan aquellos cuyas clases están fuera del rango r_i. Como en el k-NN estándar, la clase mayoritaria entre los vecinos de k se usa como la etiqueta predicha.

MkNN requiere conjuntos de datos monotónicos para funcionar correctamente. Por ello, se utiliza una técnica de re-etiquetado para transformar los datos de entrenamiento. Estas técnicas pretenden identificar las violaciones de la monotonicidad y hacer los mínimos cambios posibles con la mínima diferencia de clase para eliminarlas [4].

II-C. k-Vecinos más Cercanos Difuso

Los métodos difusos de los k-Vecinos más Cercanos [8] incluyen conceptos difusos en la clásica regla k-NN para aprender de conjuntos difusos y producir reglas de clasificación difusas. Para una nueva instancia x_i, el algoritmo original FkNN [7] extrae sus K vecinos más cercanos de la misma manera que el k-NN estándar. Sin embargo, sus pertenecías a cada clase l se calculan con la siguiente expresión:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j, l) * \frac{1}{||x - x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{1}{||x - x_j||^{(m-1)}}} \quad (2)$$

Como se muestra en 2, la pertenencia $u(x_i, l) = u_{il}$ de la muestra x_i a la clase l se asigna con el producto de la pertenencia a la clase $u(x_j, l)$ de los vecinos x_j y la inversa de sus distancias a x_i. Este último sirve como un peso que sesga hacia las pertenencias de las muestras más cercanas. El parámetro m determina el grado de influencia de las distancias a los vecinos. La etiqueta final de clase para el ejemplo x_i será aquella l con el mayor grado de pertenencia u_{il}.

Partiendo de un conjunto de entrenamiento etiquetado, FkNN [7] lo transforma en un conjunto difuso con pertenencias de clase usando la regla del vecino más cercano. Para cada muestra de entrenamiento x, se extraen sus k vecinos más cercanos y, luego, sus pertenencias se calculan de acuerdo a 3. Esta transformación ha demostrado ser útil frente a muestras ruidosas, ya que sus pertenencias se repartirán por las clases circundantes perdiendo su influencia.

$$u(x,l) = \begin{cases} 0{,}51 + 0{,}49 * (nn_l/k) & \text{if} \quad l = C(x) \\ 0{,}49 * (nn_l/k) \end{cases} \quad (3)$$

III. k-Vecinos más Cercanos Difuso para clasificación monotónica

En esta sección, se explica en detalle nuestra propuesta de k-Vecinos más Cercanos Difuso para clasificación monotónica.

III-A. De probabilidades a etiqueta final de clase

Es esencial elegir el mecanismo ideal según la monotonicidad para obtener una clase final a partir de una función de probabilidad. La clase más probable es el método más

común. Sin embargo, esto podría ser inapropiado para escenarios con restricciones monótonas. Por ejemplo, sea $x \leq y$ y sus probabilidades $f_x = (0{,}2, 0{,}2, 0{,}4, 0{,}2, 0{,}0)$ y $f_y = (0{,}0, 0{,}4, 0{,}3, 0{,}2, 0{,}1)$, entonces sus clases finales rompen la monotonía: $argmax(f_x) = l_3 > l_2 = argmax(f_y)$. Aunque, la instancia y tenga mayor peso en etiquetas más altas que x. Es más, f_y domina débilmente a f_x según el primer grado dominancia estocástica (FSD) [10], ya que la función de distribución acumulativa F_x de x, es mayor, elemento por elemento, que F_y, es decir, $f_x \preceq_{FSD} f_y \iff (\forall l \in \mathcal{Y})(F_x(l) \geq F_y(l))$. La FSD ha sido de utilidad para definir las restricciones de monotonicidad en las clasificaciones estocásticas [5], [11], con la expresión $x \leq y \implies f_x \preceq_{FSD} f_y$.

La mediana satisface dicha expresión, preservando la monotonicidad [5], [10]. Esta se calcula siguiendo la definición de percentil 50 como el rango$[l_m, l_M]$:

$$\begin{aligned} l_m &= \text{mín}\{l \in \mathcal{Y} | Prob\{X \leq l\} \geq 1/2\} \\ l_M &= \text{máx}\{l \in \mathcal{Y} | Prob\{X \geq l\} \geq 1/2\} \end{aligned} \tag{4}$$

Volviendo al ejemplo anterior, las clases para x y y elegidas por la mediana no rompen la monotonicidad: $med(f_x) = med(f_y) = 3$.

III-B. Pertenencias de clase robustas al ruido monotónico

En esta subsección, se explica el cálculo de pertenencias de clase rediseñado para reducir la influencia de las instancias no-monotónicas. En primer lugar, se ha de tratar las violaciones monótonas más simples, es decir, instancias repetidas con diferentes clases. MonFkNN primero sustituye las réplicas de cualquier ejemplo x por solo vector de característica x y sus pertenencias $u(x)$. La probabilidad $u(x, l)$ de la instancia x y la clase l se calcula con frecuencia de ejemplos duplicados pertenecientes a l, como se muestra en la siguiente expresión:

$$u(x, l) = \frac{|\{z \in \mathcal{D} | z = x \wedge C(z) = l\}|}{|\{z \in \mathcal{D} | z = x\}|} \tag{5}$$

Posteriormente, MonFkNN estima las pertenencias del resto de instancias con la información de los vecinos más cercanos a cada una. Sin embargo, estos vecinos son extraídos con un MkNN configurado como *enRango* en lugar de uno tradicional. Una vez obtenidos los vecinos de cada ejemplo x, la información de clase se fusiona en pertenencias de x. La probabilidad $u(x, l)$ de pertenecer a la clase l se calcula con la siguiente expresión:

$$u(x, l) = \begin{cases} \textit{RCr} + (nn_l/k) * (1 - \textit{RCr}) & \text{if} \quad l = C(x) \\ (nn_l/k) * (1 - \textit{RCr}) \end{cases} \tag{6}$$

donde nn_l es el número de vecinos de la clase l, k el número total de vecinos y $C(x)$ es la etiqueta original del ejemplo x. *RCr* es un nuevo parámetro llamado *Relevancia de la clase real"*. *RCr* puede ser visto como la probabilidad mínima asignada a la clase original $C(x)$ de la instancia x, en caso de que no haya vecinos etiquetados con $C(x)$.

Hay algunos valores para *RCr* en el rango $[0, 1]$ que tienen comportamientos muy interesantes y distintos. En el caso de un conjunto de datos realmente ruidoso, *RCr* podría establecerse en 0, dejando toda la responsabilidad al cálculo del rango r_i de las clases válidas y los vecinos más cercanos. En presencia de solo instancias repetidas, el usuario puede optar por tratarlas con *RCr* $= 1$. Finalmente, si se desea considerar las instancias etiquetadas originalmente, se recomienda asignar *RCr* a $0{,}5$. Este valor asegura que la clase real esté dentro del conjunto de medianas. En contraste con FkNN y sus $0{,}51$, si todos los vecinos pertenecen a la misma clase diferente a la actual, nuestro método obliga a elegir una clase intermedia. Por lo general, este último valor ofrece un comportamiento balanceado, estable y con un mejor rendimiento.

III-C. Agregación flexible de las pertenencias

Después de estimar la pertenencia de clase de cada instancia de entrenamiento, nuestro algoritmo está listo para predecir nuevos ejemplos. Esta fase de predicción ha sido diseñada con una mayor flexibilidad, permitiendo al usuario elegir entre predicciones más precisas o puramente monótonas.

Para ello, nuestro método utiliza otro MkNN para obtener los vecinos a utilizar en la predicción final. Este MkNN tiene también dos versiones, *enRango* y *fueraDeRange*. Pero son sustancialmente diferentes en comparación con las variantes originales, especialmente para la variante *fueraDeRange*.

La alternativa *enRango* se basa en la misma idea del MkNN original, donde los vecinos de un ejemplo deben pertenecer a un conjunto de clases monotónicamente válidas. Sin embargo, este rango de clases se obtiene utilizando las medianas adquiridas de las pertenencias de clase de las instancias de entrenamiento. Este avance mejora nuestro método hacia la robustez del ruido monotónico. La primera fase de MkNN se centra en la preservación de la monotonía de las medianas resultantes, para servir como una ventaja para las predicciones finales.

La versión *fueraDeRango* de nuestro método es completamente diferente de la anterior regla *fueraDeRango*. Ha sido diseñada con la intención de priorizar la precisión del clasificador sobre la monotonicidad. Con este propósito, nuestro método considera cualquier ejemplo como un vecino válido sin importar su clase. A diferencia del modelo original, no se realiza ningún filtrado o eliminación de vecinos fuera del rango válido. Sin embargo, su relevancia en la agregación de pertenencias se puede reducir, gracias a un factor de penalización introducido en la expresión de agregación.

Luego, para un nuevo ejemplo x, se obtienen sus vecinos más cercanos según la variante elegida. Sus pertenencias se agregan con la misma fórmula utilizada por el FkNN original con la adición del factor de penalización para la versión *fueraDeRange*. La siguiente expresión muestra cómo se integra este parámetro:

$$u(x,l) = \frac{\sum_{j=1}^{K} u(x_j,l) * \frac{pOR_j}{||x-x_j||^{(m-1)}}}{\sum_{j=1}^{K} \frac{pOR_j}{||x-x_j||^{(m-1)}}} \quad (7)$$

Como anteriormente, la pertenencia $u(x_i,l)$ de la nueva muestra x_i a la clase l es el resultado de la suma de las pertenencias a dicha clase $u(x_j,l)$ de los vecinos x_j ponderadas inversamente con la distancia a x_i. En la versión *fueraDeRange* de nuestro método, hay otro factor de ponderación en la contribución a las probabilidades finales, el parámetro mencionado como *"penalización de fueraDeRango"* (*pOR*). El factor pOR_j sólo se aplica si la clase y_j del vecino x_j no se encuentra en el rango de clases válidas r_i de x_i. Se puede configurar con valores continuos de 0 a 1. Cuando se asigna a 1, no se aplica ninguna penalización. El valor 0 significa una penalización completa, es decir, los vecinos con clases inválidas no participarán en la agregación. Este último comportamiento es equivalente a la variante *fueraDeRango* del MkNN original. Se recomienda el uso de 0.5, ya que es un buen equilibrio entre reducir su relevancia y considerarlos en la decisión.

Finalmente, la predicción de clase del nuevo ejemplo x es la mediana de las pertenencias de clase normalizadas resultantes.

MonFkNN se ha desarrollado para ser robusto al ruido monotónico y versátil en múltiples escenarios gracias a sus parámetros. Entre sus posibilidades, destacan dos configuraciones con un comportamiento muy distintivo: Monotónico Puro (PM) y Monotónico Aproximado (AM). La configuración monotónica pura corresponde a un valor de 0.5 para el parámetro *RCr* y el uso de la regla *enRango* para obtener las pertenencias de nuevas instancias. Este enfoque pretende dar predicciones con las mínimas violaciones de la monotonicidad.

La configuración monotónica aproximada prioriza la capacidad de predicción y relaja las restricciones monótonas. Las pertenencias del entrenamiento se obtienen mediante el tratamiento de las muestras repetidas. Los ejemplos únicos tendrán una probabilidad de 1 para la clase real y 0 para el resto. Este comportamiento se consigue con $RCr = 1$. Ya que buscamos predicciones más precisas, se consideran todas las instancias como vecinos válidos. Sin embargo, aquellas con etiquetas de clase inválidas contribuirán con la mitad de sus probabilidades ($pOR = 0{,}5$).

IV. Metodología experimental

Esta sección introduce el marco experimental utilizado en los diferentes estudios empíricos del trabajo. En dichos experimentos, se han incluido 12 conjuntos de datos diferentes. Estos son retratados en la Tabla I, donde se detallan el número de instancias, atributos y clases de cada conjunto en las columnas Ins., At. y Cl., respectivamente. La columna Direcciones indica la dirección de la relación monotónica entre cada atributo y la clase: monotonía directa (+) o inversa (-).

Se ha llevado a cabo un esquema de validación cruzada de 10 particiones (10-fcv) para ejecutar los diferentes clasificadores sobre estos conjuntos. Sus particiones han sido extraídas del repositorio KEEL [12].

Tabla I: Descripción de los 12 conjunto de datos usados.

Conjuntos	Ins.	At.	Cl.	Direcciones
artiset	1000	2	10	directas
balance	625	4	3	{-, -, +, +}
bostonhousing4cl	506	13	4	{-, +, -, +, -, +, -, +, -, -, -, +, -}
car	1728	6	4	directas
ERA	1000	4	9	directas
ESL	488	4	9	directas
LEV	1000	4	5	directas
machineCPU	209	6	4	{-, +, +, +, +, +}
qualitative_bankruptcy	250	6	2	directas
SWD	1000	10	4	directas
windsorhousing	546	11	2	directas
wisconsin	683	9	2	directas

La tabla II muestra los clasificadores involucrados en la comparación experimental y sus parámetros.

Tabla II: Parámetros considerados para los algoritmos a comparar.

Algoritmos	Parámetros
MkNN [4]	k = 5, distancia = euclidiana, tipoVecinos = enRango
OSDL [5]	equilibrado = No, tipoClasificación = mediana, límiteInferior = 0, límiteSuperior = 1 ajusteInterpolación = No, ponderado = No, pasoInterpolación = 10, gradoInterpolación = 0.5
OLM [6]	resolución = conservativa tipoClasificación = conservativa
MID [3]	R = 1, confianza = 0.25, elementosPorHoja = 2
Mon Fuzzy$k$NN	k = 5, K = 9, distancia = euclidiana
Monotónico Puro	RCr = 0.5, tipoVecinos = enRango
Monotónico Aprox.	$RCr = 1$, tipoVecinos = fueraDeRango, $pOR = 0{,}5$

Para evaluar la competencia de los clasificadores, se han empleado tres medidas que cubren diferentes aspectos de su rendimiento: capacidad predictiva con la precisión estándar, coste de los errores con el error absoluto medio (MAE) y monotonicidad con el Índice No Monotónico (NMI). El NMI mide la proporción de pares de muestras que rompen la monotonicidad respecto al total de pares.

V. Resultados y análisis

Esta sección presenta los resultados de diferentes estudios empíricos y sus análisis.

V-A. *Evaluación de las propuestas de k-Vecinos más Cercanos Difuso Monotónico*

A continuación, una comparación entre las versiones Pura y Aproximada de MonFkNN, destacando los diferentes comportamientos y aspectos de su rendimiento. La Tabla III contienen los resultados de las dos configuraciones de la propuesta en términos de Precisión, MAE y NMI. La fuente en negrita indica los mejores resultados obtenidos para cada conjunto de datos y métrica.

En la Tabla III, las diferencias entre ambos enfoques quedan claramente patentes. Tal como fueron diseñados, MonFkNN Aproximado (AM) tiene una mejor precisión en promedio en

Tabla III: Resultados de las versiones Puramente y Cuasi Monotónicas de MonFkNN.

	Precisión		MAE		NMI	
	PM ($RCr = 0,5$)	**AM** ($pOR = 0,5$)	**PM** ($RCr = 0,5$)	**AM** ($pOR = 0,5$)	**PM** ($RCr = 0,5$)	**AM** ($pOR = 0,5$)
artiset	0,9309	**0,9349**	0,0691	**0,0651**	**0,0000**	**0,0000**
balance	**0,9307**	0,9008	**0,0853**	0,1168	**0,0000**	0,0001
bostonhousing4cl	0,6561	**0,7134**	0,3972	**0,3261**	**0,0000**	0,0001
car	0,9740	**0,9834**	0,0295	**0,0195**	**0,0000**	**0,0000**
ERA	0,2420	**0,2430**	**1,2813**	1,2993	**0,0052**	**0,0052**
ESL	0,7036	**0,7131**	0,3149	**0,3053**	0,0004	**0,0003**
LEV	**0,6377**	0,6110	**0,3927**	0,4223	**0,0004**	0,0009
machineCPU	**0,7033**	0,6699	**0,3158**	0,3493	**0,0002**	0,0017
qualitative_bankruptcy	**0,9960**	**0,9960**	**0,0040**	**0,0040**	**0,0000**	**0,0000**
SWD	0,5807	**0,5833**	**0,4370**	0,4380	0,0007	**0,0003**
windsorhousing	0,7576	**0,7839**	0,2424	**0,2161**	**0,0005**	0,0051
wisconsin	0,9653	**0,9663**	0,0347	**0,0337**	**0,0000**	**0,0000**
Media:	0,7565	**0,7583**	0,3003	**0,2996**	**0,0006**	0,0012

el 75 % de los conjuntos utilizados. Por otro lado, MonFkNN Puro (PM) logra predicciones monotónicamente más confiables en 10 de los 12 conjuntos de datos utilizados, con grandes diferencias en los problemas de *Windsorhousing* y *MachineCPU*. Ambos tienen un resultado estable y bueno en términos de MAE, siendo la versión AM un poco mejor.

Ya que la monotonicidad se priorizada la mayoría de las veces en la clasificación con restricciones monotónicas, usaremos la variante Pura de MonFkNN en los siguientes estudios empíricos.

V-B. Comparación con el Estado-del-Arte

A lo largo de este experimento, se evalúa el rendimiento de MonFkNN en comparación a los clasificadores monotónicos del Estado-del-Arte. En esta, se busca un equilibrio entre predicciones precisas y monotónicas.

La Tabla IV recoge los resultados de precisión para los diferentes conjuntos de datos obtenidos por los algoritmos probados. En términos de precisión, MonFkNN rinde bastante mejor con un amplio margen. Nuestro método logra predicciones más precisas para 7 conjuntos de datos, con casos particularmente notables, como *balance* o *ESL*.

Tabla IV: Resultados en precisión obtenidos por los algoritmos evaluados.

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MID**
artiset	**0,9309**	0,9199	0,1952	0,7948	0,7237
balance	**0,9307**	0,8624	0,6352	0,8320	0,7808
bostonhousing4cl	0,6561	0,6126	0,2787	0,5277	**0,6739**
car	**0,9740**	0,9711	0,9549	0,9543	0,8027
ERA	0,2420	0,1990	0,2320	0,1690	**0,2760**
ESL	**0,7036**	0,6332	0,6721	0,5738	0,6414
LEV	0,6377	0,4630	**0,6400**	0,4250	0,6070
machineCPU	**0,7033**	0,6890	0,2919	0,6746	0,6220
qualitative_bankruptcy	**0,9960**	**0,9960**	0,9160	0,9800	0,9840
SWD	0,5807	0,5200	**0,5840**	0,4160	0,5540
windsorhousing	0,7576	0,5861	0,4927	0,7564	**0,8205**
wisconsin	**0,9653**	0,9649	0,9590	0,8873	0,9517
Media:	**0,7565**	0,7014	0,5710	0,6659	0,7031

En la clasificación con restricciones monotónicas, el coste de los errores puede ser esencial. La Tabla V muestra el error en forma de MAE cometido por cada clasificador. Al igual que el rendimiento de precisión, MonFkNN es claramente mejor que el resto con el error más pequeño en promedio y para 8 de los conjuntos de datos. En aquellos problemas donde otros son superiores, como *LEV* o *wisconsin*, este consigue resultados realmente cercanos.

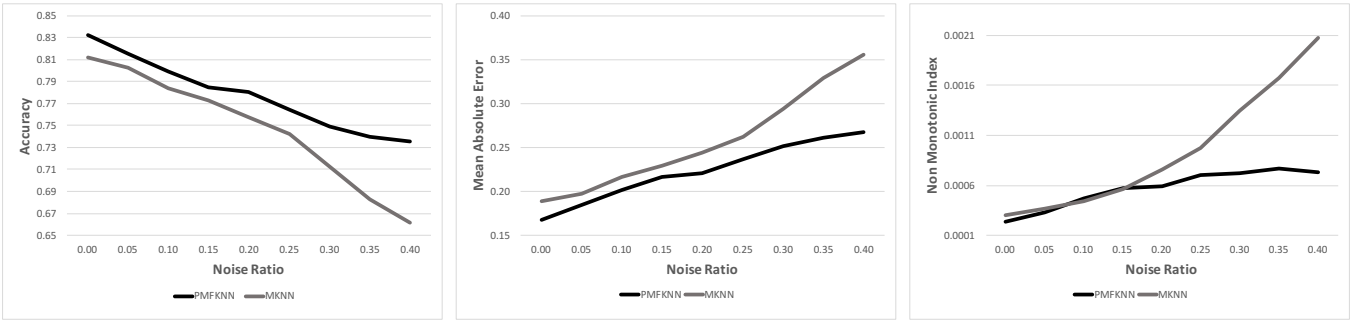
Tabla V: Resultados en MAE obtenidos por los algoritmos evaluados.

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MID**
artiset	**0,0691**	0,0771	1,6897	0,2082	0,3123
balance	**0,0853**	0,1504	0,4912	0,1920	0,3360
bostonhousing4cl	0,3972	0,4901	0,9368	0,5988	**0,3893**
car	**0,0295**	0,0359	0,0475	0,0538	0,2506
ERA	**1,2813**	1,4270	1,2850	2,1500	1,2970
ESL	**0,3149**	0,3791	0,3607	0,4734	0,3934
LEV	0,3927	0,5740	**0,3920**	0,6680	0,4290
machineCPU	**0,3158**	0,3301	0,9043	0,3589	0,4211
qualitative_bankruptcy	**0,0040**	**0,0040**	0,0840	0,0200	0,0160
SWD	**0,4370**	0,4840	**0,4370**	0,7630	0,4750
windsorhousing	0,2424	0,4304	0,5073	0,2436	**0,1795**
wisconsin	0,0347	**0,0337**	0,0410	0,1127	0,0483
Media:	**0,3003**	0,3680	0,5980	0,4869	0,3790

En relación a la monotonicidad, la Tabla VI presenta los resultados de los NMI alcanzados. En este caso, la competencia por ser el mejor está mucho más reñida. OLM y MID obtienen predicciones claramente menos monótonas. Este último tiene el peor comportamiento considerando sólo la monotonicidad. MonFkNN, MkNN y OSDL funcionan de manera similar. MonFkNN y OSDL son ligeramente mejores en promedio.

Tabla VI: Resultados en NMI obtenidos por los algoritmos evaluados.

	MonFkNN-PM	**MkNN**	**OSDL**	**OLM**	**MID**
artiset	**0,0000**	**0,0000**	**0,0000**	**0,0000**	0,0039
balance	**0,0000**	0,0001	0,0006	**0,0000**	0,0017
bostonhousing4cl	**0,0000**	**0,0000**	**0,0000**	0,0003	0,0022
car	**0,0000**	**0,0000**	**0,0000**	**0,0000**	0,0046
ERA	0,0052	0,0056	**0,0049**	0,0063	0,0082
ESL	**0,0004**	0,0012	0,0006	0,0025	0,0021
LEV	**0,0004**	0,0010	**0,0004**	0,0043	0,0018
machineCPU	0,0002	**0,0000**	**0,0000**	0,0014	0,0037
qualitative_bankruptcy	**0,0000**	**0,0000**	0,0003	**0,0000**	0,0002
SWD	0,0007	**0,0005**	0,0009	0,0015	0,0020
windsorhousing	0,0005	**0,0000**	**0,0000**	**0,0000**	0,0030
wisconsin	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
Media:	**0,0006**	0,0007	**0,0006**	0,0014	0,0028



(a) Efecto del ruido en la Precisión. (b) Efecto del ruido en el MAE. (c) Efecto del ruido en el NMI.

Figura 1: Rendimiento de Puro MonFkNN y MkNN con los diferentes conjuntos ruidos de *Artiset*.

V-C. Sobre la robustez al ruido monotónico de k-Vecinos más Cercanos Difuso Monotónico

Con este último estudio empírico, pretendemos probar la robustez de MonFkNN frente a la presencia de violaciones monótonas o ruido en los sets de entrenamiento en contraste con MkNN. Para ello, se ha introducido diferentes cantidades de instancias ruidosas en las particiones de entrenamiento del conjunto de datos artificial *Artiset*. Luego, se han comparado el comportamiento de MonFkNN y MkNN en términos de precisión, MAE y NMI, mientras que el ratio de ruido aumenta.

La figura 1 muestra el comportamiento de MonFkNN y MkNN (líneas más oscuras y más claras, respectivamente) en base a la precisión (1a), MAE (1b) y NMI (1c), con la progresión del ruido. Como era de esperar, mientras la cantidad de ruido aumenta, el rendimiento de ambos métodos empeora. Sin embargo, hay alguna gran diferencia entre ambos clasificadores.

En primer lugar, el comportamiento de cara al ruido de MonFkNN es claramente mejor que el del MkNN en todos los aspectos probados. Las líneas negras siempre están sobre las más claras de la figura 1a, lo que indica siempre una precisión mayor, y debajo de ellas en las figuras 1b y 1c, lo que significa mejores MAE y NMI para MonFkNN. Por lo general, la distancia entre ambos métodos es amplia, con excepción de los resultados de NMI obtenidos para los valores más pequeños de ruido. Además, mientras que el índice de ruido aumenta, sus diferencias también aumentan.

Con estos experimentos, MonFk-NN ha demostrado una fuerte robustez al ruido monotónico preservando su buen rendimiento en términos de precisión, costes de error y monotonicidad.

VI. Conclusiones

En este trabajo, se ha propuesto un modelo difuso de k-Vecinos más Cercanos para la clasificación con restricciones monotónicas. La transferencia de una función de probabilidades a una clase final se ha adaptado para que respete dichas restricciones. Se ha diseñado mecanismos para reducir la influencia de las violaciones monotónicas. Como demostración de su flexibilidad, se han presentados dos configuraciones diferentes del modelo con comportamientos muy distintos.

Durante el análisis experimental, se ha mostrado el gran potencial en relación a la monotonicidad de la configuración puramente monotónica y la buena precisión de la versión aproximada de MonFkNN. En comparación con otros métodos, MonFkNN es significativamente mejor en términos de precisión y coste de error, igualando los mejores resultados de NMI. Además, ha demostrado una fuerte robustez a grandes cantidades de ruido preservando su buen rendimiento.

Referencias

[1] C.-C. Chen and S.-T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7235–7247, 2014.

[2] J.-R. Cano, N. R. Aljohani, R. A. Abbasi, J. S. Alowidbi, and S. Garcia, "Prototype selection to improve monotonic nearest neighbor," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 128–135, 2017.

[3] A. Ben-David, "Monotonicity maintenance in information-theoretic machine learning algorithms," *Machine Learning*, vol. 19, no. 1, pp. 29–43, 1995.

[4] W. Duivesteijn and A. Feelders, "Nearest neighbour classification with monotonicity constraints," in *ECML/PKDD (1)*, 2008, pp. 301–316.

[5] S. Lievens, B. De Baets, and K. Cao-Van, "A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting," *Annals of Operations Research*, vol. 163, no. 1, pp. 115–142, 2008.

[6] A. Ben-David, "Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications," *Decision Sciences*, vol. 23, pp. 1357–1372, 1992.

[7] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[8] J. Derrac, S. García, and F. Herrera, "Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects," *Information Sciences*, vol. 260, pp. 98–119, 2014.

[9] W. Kotłowski and R. Słowiński, "On nonparametric ordinal classification with monotonicity constraints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2576–2589, 2013.

[10] H. Levy, *Stochastic dominance: Investment decision making under uncertainty*. Springer, 2015.

[11] S. Lievens and B. De Baets, "Supervised ranking in the weka environment," *Information Sciences*, vol. 180, no. 24, pp. 4763–4771, 2010.

[12] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesús, L. Sánchez, and F. Herrera, "Keel 3.0: an open source software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 1238–1249, 2017.