E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Implementación del algoritmo de los k vecinos más cercanos (k-NN) y estimación del mejor valor local de k para su cálculo

Grado en Ingeniería Informática

Trabajo Fin de Grado

Gonzalo Berástegui Arbeloa
Mikel Galar Idoate
Pamplona, 22 de Marzo 2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

Éste trabajo se centrará en los problemas de clasificación. El objetivo del mismo es el de implementar el algoritmo de los k vecinos más cercanos (k-NN) pero con un método distinto al original. La idea es obtener un valor local de k para cada ejemplo del conjunto de entrenamiento, éste valor lo obtendremos comprobando el rendimiento de todos los valores de k dentro de un intervalo [kmin, kmax] para cada ejemplo. Una vez que tengamos el valor local de k asociado a cada ejemplo del conjunto de entrenamiento, clasificaremos el conjunto de test mediante k-NN utilizando con cada uno de ellos el valor local de k que tenga asignado el ejemplo más próximo del conjunto de entrenamiento. También realizaremos una comparativa entre éste método y otros conocidos para ver si es mejor.

El método se ha implementado siguiendo el siguiente artículo:
A Proposal for Local k Values for k-Nearest Neighbor Rule
Author(s): Nicolás García-Pedrajas; Juan A. Romero del Castillo; Gonzalo Cerruela-García
Page(s): 470 – 475
http://ieeexplore.org/document/7368188/

Materias o Palabras clave: clasificación, k vecinos más cercanos, minería de datos, aprendizaje automático.

Índice

1. Introducción....4
1.1 Sistemas de clasificación....4
1.2 Algoritmo k-NN....4
1.3 Objetivos....5
2. Preliminares....6
2.1 Profundizando en k-NN....6
2.1.1 Normalización....6
2.1.2 Distancias....6
2.1.3 Medidas de rendimiento....7
3. Desarrollo e Implementación de k-NN local....9
3.1 Explicación del algoritmo k-NN local....9
3.2 Implementación del algoritmo K-NN local....10
3.2.1 Accuracy global....11
3.2.2 Accuracy local....12
3.2.3 Accuracy total o eval....15
3.2.4 Clasificar nuevos ejemplos....15
4. Marco Experimental....17
4.1 Métodos a comparar....17
4.2 Datasets....17
4.3 Medidas y test estadísticos....22
4.3.1 Medidas....22
4.3.2 Test estadísticos....23
5. Estudio Experimental....24
5.1 Resultados....24
5.1.1Conjunto “small”....25
5.1.2 Conjunto “big”....29
5.1.3 Conjunto “imb1”....31
5.2 Gráficas....35
5.2.1 Gráficas del conjunto “small”....36
5.2.2 Gráficas del conjunto “big”....37
5.2.3 Gráficas del conjunto “imb1”....38
5.2.4 Gráficas del conjunto “imb2”....38
5.3 Test estadísticos....39
5.3.1 Test para el conjunto “small”....39
5.3.2 Test para el conjunto “big”....40
5.3.3 Test para el conjunto “imb1”....41
5.3.4 Test para el conjunto “imb2”....42
5.3.5 Tablas de tiempos....43
5.4 Discusión....43
6. Conclusiones y Líneas Futuras....49
7. Referencias....50

1. Introducción

1.1 Sistemas de clasificación

Existen cantidades gigantescas de datos almacenados en bases de datos, data warehouses y otros tipos de almacenes de información. Esta explosión de información ha provocado la aparición del KDD (Knowledge Discovery from Databases) que es el proceso completo de extracción de conocimiento a partir de bases de datos. Dentro del KDD existe una etapa conocida como Minería de Datos, el objetivo de esta etapa es crear un modelo basado en los datos que tenemos con el que podamos hacer predicciones, entender mejor los datos o explicar situaciones pasadas.

El aprendizaje para crear el modelo puede ser de distintos tipos:

- **Aprendizaje supervisado**: aprendemos a partir de un conjunto de instancias pre-etiquetadas un método parar predecir instancias sin etiquetar.

- **Aprendizaje no supervisado**: no hay conocimiento a priori sobre el problema, es decir, no hay instancias etiquetadas. El método debe ser capaz de encontrar alguna estructura en los datos que permita organizarlos de algún modo.

Los sistemas de clasificación son un ejemplo de aprendizaje supervisado. Un sistema de clasificación es aquel que clasifica en diferentes clases un conjunto de ejemplos desconocidos a partir de otro conjunto de ejemplos de los cuales ya se conoce su clase. Para ello, se construye un modelo llamado clasificador que nos permite predecir la clase de cualquier nuevo ejemplo.

Todo sistema de clasificación tiene 2 etapas:

- **Aprendizaje**: la construcción del modelo. El modelo se construye a partir de un conjunto de ejemplos ya clasificados. El modelo obtenido se representa como un conjunto de reglas de clasificación, árboles de decisión, fórmula matemática, etc.

- **Validación**: estimación de la precisión del modelo. Se prueba el modelo con un conjunto de ejemplos diferentes al utilizado para la construcción del modelo. Para cada ejemplo se compara su clase real con la clase predicha por el clasificador, el ratio de precisión es el porcentaje de ejemplos que el modelo clasifica correctamente.

1.2 Algoritmo k-NN

El algoritmo k-NN es uno de los algoritmos de clasificación más conocidos y un ejemplo de aprendizaje supervisado. En este algoritmo usaremos un conjunto de ejemplos ya clasificados a los que llamaremos conjunto de entrenamiento o train para clasificar los nuevos ejemplos. No se crea un nuevo modelo, sino que el modelo es el propio conjunto de train.

El algoritmo se llama así k-NN (k-Nearest Neighbors) porque clasifica cada nuevo ejemplo calculando la distancia de ese ejemplo con todos los del conjunto de train. La clase predicha para este nuevo ejemplo vendrá dada por la clase a la que pertenezcan los ejemplos más cercanos del conjunto de train, el valor de la k es el que determina en cuantos vecinos debemos fijarnos para predecir la clase. Así, con un valor de k = 1, la clase predicha para cada nuevo ejemplo será la clase a la que pertenezca el ejemplo más cercano del conjunto de train.

El principal problema del algoritmo k-NN es encontrar el valor de k con el que obtengamos un mayor rendimiento al clasificar, generalmente se utiliza una técnica conocida como cross validation. Ésta técnica consiste en dividir el conjunto de train en distintas partes, por ejemplo en 5 y clasificar cada una de esas partes mediante k-NN con los valores de k que se quieran, utilizando las otras 4 partes como conjunto de train. De este modo tendremos el rendimiento de k-NN con todos los valores de k para cada una de las 5 particiones que hemos hecho del conjunto de train original. Haciendo la media de esos rendimientos obtendríamos el valor de k con el que mejor rendimiento se obtiene y lo utilizaríamos para clasificar cualquier nuevo ejemplo.

Sin embargo existe otra técnica que no clasifica con el mismo valor de k cada nuevo ejemplo, sino que la k con la que clasificaremos a un nuevo ejemplo vendrá determinada en función de distintos parámetros como por ejemplo cuáles son sus vecinos más cercanos. El proyecto que vamos a realizar utiliza esta otra técnica.

1.3 Objetivos

El objetivo de este trabajo es el de implementar una variante del algoritmo k-NN estándar llamada k-NN local y comprobar si es más eficiente que el método estándar u otros contra los que lo enfrentaremos.

En este nuevo método no habrá un valor de k general para todo el modelo con el que predecir los nuevos ejemplos, sino que a cada ejemplo del conjunto de train se le asigna un valor de k local que sea óptimo para clasificar su vecindario. Cada nuevo ejemplo se clasificará usando el algoritmo k-NN pero con ése valor local de k que tenga su vecino más cercano del conjunto de train.

2. Preliminares

2.1 Profundizando en k-NN

El algoritmo k-NN (k-Nearest Neighbors) como ya hemos comentado es uno de los algoritmo de clasificación más conocidos que hay. Este algoritmo clasifica un conjunto de ejemplos a partir de otros de los que ya conocemos su clase y a los que llamaremos conjunto de entrenamiento o train. Lo que hace es calcular la distancia de cada ejemplo a clasificar con todos los ejemplos del conjunto de train y clasifica al ejemplo en función de la clase a la que pertenezcan aquellos ejemplos más cercanos. La variable k se utiliza para determinar cuántos de sus ejemplos más cercanos del conjunto de train han de tenerse en cuenta para clasificarlo.

2.1.1 Normalización

Lo primero que hace el algoritmo k-NN es calcular la distancia de cada ejemplo nuevo a clasificar con todos los del conjunto de train. Los ejemplos tienen una serie de atributos, donde cada atributo tiene un rango de valores que puede ser muy distinto del rango de valores de otro atributo, con lo cual a la hora de calcular las distancias, un atributo con un rango de valores muy alto podría solapar a todos los demás atributos. Esto hace necesario normalizar todos los atributos para que su rango de valores esté entre 0 y 1 de forma que todos los atributos afectan por igual al cálculo de la distancia. Para ello, a cada atributo le restamos el valor mínimo que puede tener ese atributo y lo dividimos por la diferencia entre el valor máximo y mínimo que puede tener ese atributo.

$$e_k^j = \frac{e_k^j - min^j}{max^j - min^j}, con\ k = \{1, \dots, P\}$$

2.1.2 Distancias

Una vez que todos los atributos tienen el mismo rango de valores podemos medir la distancia del nuevo ejemplo a clasificar con todos los del conjunto de train. Para calcular la distancia disponemos de distintos métodos:

- **Distancia Euclídea**: la que se utiliza de manera predeterminada y que usaremos en nuestro estudio experimental. La distancia entre 2 puntos es la línea recta que los une.

$$d_e(e_i, e') = \sqrt{\sum_{j=1}^{N}(e_i^j - e'^j)^2}$$

- **Distancia Manhattan**: la distancia entre dos puntos es la suma de las diferencias absolutas entre sus coordenadas.

$$d_m(e_i, e') = \sum_{j=1}^{N}\left|e_i^j - e'^j\right|$$

- **Distancia Canberra**: similar a la distancia Manhattan pero más receptiva a los puntos de datos más cercanos al origen.

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

- **Distancia Máximo**: La distancia entre dos puntos se calcula como la diferencia mayor en cualquiera de las dimensiones de coordenadas.

2.1.3 Medidas de rendimiento

Una vez que hemos clasificado los nuevos ejemplos en base a la clase de sus vecinos más cercanos del conjunto de train, debemos ser capaces de medir el rendimiento del algoritmo k-NN. No siempre se utiliza la misma medida de rendimiento, dependerá de lo que busquemos: precisión, velocidad, robustez, escalabilidad, etc. Si lo que buscamos es precisión hay distintos tipos de medidas de rendimiento para calcularla. Unas están basadas en una matriz de confusión y otras en las curvas ROC. Nos centraremos en las basadas en una matriz de confusión.

Dado un conjunto de ejemplos a clasificar en m clases, su matriz de confusión es una matriz m x m donde cada MC(i,j) es el número de ejemplos que se han clasificado como j, cuando su clase correcta es i.

		Clasificación como			
		1	2	…	m
Clase real	1	real1clas1	real1clas2	…	real1clasm
	2	real2clas1	real2clas2	…	real2clasm
	…	…	…	…	…
	m	realmclas1	realmclas2	…	realmclasm

A partir de esta matriz se pueden obtener muchas medidas de rendimiento:

- **Accuracy**: el número de ejemplos clasificados correctamente.

$$Acc = \frac{(real1\,clas1 + real2\,clas2 + ... + realmclasm)}{total}$$

- **Precisión por clase (Pi)**: el número de ejemplos clasificados correctamente de una sola clase.

$$P_2 = \frac{real2\,clas2}{(real2clas1 + real2\,clas2 + ... + real2\,clasm)}$$

- **Accuracy medio por clases**: la media de la precisión por clase de todas las clases.

$$AccAvg = \frac{(P_1 + P_2 + ... + P_m)}{m}$$

- **Media Geométrica**: la raíz m-ésima del producto de la precisión por clase de todas las clases.

$$GM = \sqrt[m]{P_1 * P_2 * ... * P_m}$$

- **Medida k de Cohen**: es una medida que compensa los aciertos aleatorios. Mide la precisión de la clasificación, es un método muy útil para medir la precisión de un clasificador. El valor de k se calcula con la siguiente fórmula:

$$\kappa = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{i.} x_{.i}}{n^2 - \sum_{i=1}^{C} x_{i.} x_{.i}}$$

  Donde n es el número de ejemplos, C el número de clases, $x_{i.}$ es toda la fila para el valor de i, $x_{.i}$ es toda la columna para el valor de i y x_{ii} es el valor de la celda de la diagonal principal para ese valor de i.

  El valor de k obtenido estará entre -1 (total discrepancia) y 1 (total coincidencia).

3. Desarrollo e Implementación de k-NN local

En esta sección vamos a explicar en qué consiste exactamente esta variante del k-NN estándar que hemos implementado. Explicaremos minuciosamente cada uno de los pasos que da el algoritmo con el que lo hemos implementado mediante un pequeño ejemplo.

3.1 Explicación del algoritmo k-NN local

Éste método que vamos a implementar consiste en obtener un valor de k local asociado a cada ejemplo del conjunto de train (a partir de ahora llamaremos prototipo a cada ejemplo del conjunto de train). El método aprende éste valor a partir del propio conjunto de train. Para cada prototipo, evaluamos el rendimiento de k-NN con todos los valores de k en un intervalo $[k_{min}, k_{max}]$ y escogemos el de mejor valor.

El enfoque del método es el siguiente, asignaremos a cada prototipo un valor de k, cuyo valor ideal será el número óptimo de vecinos a tener en cuenta para clasificar un ejemplo que esté en su vecindario (que un ejemplo esté en el vecindario de un prototipo significa que ése prototipo es su vecino más cercano). En el método k-NN estándar los prototipos son de la forma:

$$(x_1,...,x_n, y)$$

Donde $x_1,...,x_n$ son los atributos del prototipo e y es la clase a la que pertenece. Sin embargo, con éste método modificaremos los prototipos para que sean de la forma:

$$(x_1,\ldots,x_n, y, k)$$

Donde k será el valor local de k asociado al prototipo que utilizaremos para clasificar los ejemplos de su vecindario.

Para obtener el valor local óptimo de k de cada prototipo necesitaremos 2 medidas distintas:

- **Accuracy Global**: esta medida se calculará ejecutando el algoritmo k-NN estándar sobre todo el conjunto de train, con todos los valores de k desde 1 hasta k_{max}. Para ello, haremos una validación cruzada de 10 particiones sobre el conjunto de train y ejecutaremos el algoritmo k-NN estándar para clasificar cada una de esas 10 particiones utilizando como clasificador las otras 9. Con ello, obtendremos la precisión del algoritmo k-NN en cada una de las 10 particiones, haciendo la media obtendremos un vector al que llamaremos acc_{global} que será una estimación de la precisión del algoritmo k-NN estándar con todos los valores de k para el conjunto de train.

- **Accuracy Local**: para calcular esta medida también ejecutaremos el algoritmo k-NN estándar sobre todo el conjunto de train, pero ésta vez nos fijaremos en el efecto de cada valor de k desde 1 hasta k_{max} en el vecindario de cada prototipo. En principio, el vecindario de cada prototipo son sólo aquellos prototipos para los cuales éste es el vecino más cercano. Sin embargo, algunos prototipos no son el vecino más cercano

de ningún otro o de muy pocos. Para evitar el efecto negativo de que haya muy pocos prototipos en un vecindario, el vecindario de cada prototipo estará formado por aquellos prototipos para los cuales éste sea uno de los 3 vecinos más cercanos. Así nos aseguramos vecindarios de tamaño mayor.

El resultado será una matriz $n_{prototipos}$ x k_{max} a la que llamaremos acc_{Local}, donde tendremos la precisión del k-NN estándar sobre el vecindario de cada prototipo con todos los valores de k.

Ahora tenemos una medida que contiene la precisión del algoritmo k-NN sobre todo el conjunto de train y otra que contiene la precisión del algoritmo k-NN sobre el vecindario de cada prototipo. Necesitamos tener en cuenta ambas medidas para obtener el valor óptimo de k que irá asociado a cada prototipo, por lo que calcularemos una nueva medida llamada eval donde tendremos ambas medidas. Para ello, a la precisión que teníamos del k-NN sobre cada prototipo le vamos a sumar la precisión del k-NN sobre todo el conjunto de entrenamiento, es decir, a cada fila de la matriz acc_{local} le sumaremos el vector acc_{global}:

$$eval = acc_{local} + acc_{global}$$

Esta nueva medida eval será una matriz $n_{prototipos}$ x k_{max}, donde tendremos la precisión total del algoritmo k-NN sobre todo el conjunto de train. Cada fila representa un prototipo, por tanto, la posición cuyo valor de precisión sea más alto de cada fila, será el valor local de k asociado a ese prototipo.

Con el valor local de k ya calculado y añadido a cada prototipo, sólo quedaría clasificar nuevos ejemplos. Calculamos para cada nuevo ejemplo la distancia a todos los prototipos y clasificamos el ejemplo teniendo en cuenta para ello tantos vecinos como indique la k local de su prototipo más cercano.

3.2 Implementación del algoritmo K-NN local

El algoritmo que vamos a implementar se muestra en la figura 1, ha sido sacado del artículo: N. García-Pedrajas, J. A. Romero del Castillo and G. Cerruela-García, "*A Proposal for Local k Values for k-Nearest Neighbor Rule*," in IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 2, pp. 470-475, Feb. 2017. [1]

Algorithm 1 Outline of the Proposed Algorithm

Data : A training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^D$, a minimum value of k, k_{min}, and a maximum value of k, k_{max}.

Result : The vector of local k values $\mathbf{k}$.

1 Obtain vector of 10-fold cross-validation accuracy for global values of k from k_{min} to k_{max}.

for *every* $x \in T$ **do**

  for $k = k_{min}$ *to* $k = k_{max}$ **do**

2 Obtain $eval(k)$ using eq. 1

  end

3 Obtain optimal local value of k

4 Assign optimal value of k to k_i

end

5 Return $\mathbf{k}$

Figura 1. Algoritmo k-NN local

Hemos dividido la implementación del algoritmo en 4 partes bien diferenciadas: el cálculo del accuracy global, el cálculo del accuracy local, el cálculo de la k local y por último la clasificación de nuevos ejemplos. Para que quede claro lo que hace el algoritmo que vamos a implementar, ilustraremos cada uno de los pasos con un pequeño conjunto de train (véase tabla 1) como ejemplo. En la figura 2 se muestran los ejemplos dibujados en una gráfica, representados con círculos los ejemplos de la clase 1 y con equis los de la clase 2.

x1	x2	y
0.3	0.6	1
0.2	0.8	1
0.5	0.5	2
0.2	0.6	1
0.7	0.8	2
0.3	0.9	1
0.7	0.5	2

Tabla 1. Conjunto de train

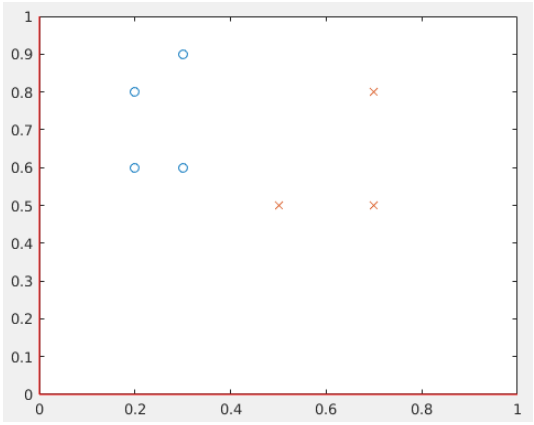


Figura 2. Gráfica del conjunto de train

3.2.1 Accuracy global

Como hemos dicho, para el cálculo del accuracy global debemos aplicar k-NN sobre todo el conjunto de train y obtener su precisión para todos los valores de k. Hacemos validación cruzada de 10 particiones, es decir, dividimos el conjunto de train en 10 particiones distintas, todas ellas con un número balanceado de ejemplos de cada clase y aplicamos el algoritmo k-NN para clasificar cada una de esas 10 partes, utilizando como clasificador las otras 9. La media de estas 10 medidas de precisión que obtendremos será el accuracy global.

Como el conjunto de train que tenemos de ejemplo no es lo suficientemente grande como para hacer 10 particiones distintas lo haremos de otra forma. Cada prototipo lo tomaremos como una partición y lo clasificaremos utilizando para ello el resto de prototipos, es decir, un leave-one-out. Obtendremos la precisión del k-NN con cada prototipo, la media de todos ellos será nuestro accuracy global:

$$acc_{Global} = [\ 100\ 85.71\ 85.71\ 57.14\ 57.14\ ]$$

Se puede ver que con k = 1 clasificamos todos correctamente, pero que a partir de k = 2 comienzan los errores al clasificar. En nuestra implementación, donde sí que hacemos validación cruzada de 10 particiones, obtendríamos 10 vectores distintos, la media de todos ellos por columnas sería el accuracy global.

3.2.2 Accuracy local

El cálculo del accuracy local es algo más complejo y difícil de ver, mostraremos todos los pasos que hay que hacer.

Primero calcularemos la distancia de cada prototipo con todos los demás y a partir de esta medida obtendremos 2 matrices distintas. Una, llamada vecinos (véase tabla 2), contendrá para cada prototipo, sus 3 vecinos más cercanos. La otra, llamada vecinosClase (véase tabla 3), contendrá para cada prototipo, la clase de sus k_{max} vecinos más cercanos en orden de cercanía. Obviamente cada prototipo es el vecino más cercano de sí mismo, pero no lo tenemos en cuenta ya que haría que el valor óptimo de k fuese 1 siempre.

Prototipo 1	4	3	2
Prototipo 2	6	4	1
Prototipo 3	7	1	4
Prototipo 4	1	2	3
Prototipo 5	7	3	6
Prototipo 6	2	1	4
Prototipo 7	3	5	1

Tabla 2. Matriz vecinos

Prototipo 1	1	2	1	1	2
Prototipo 2	1	1	1	2	2
Prototipo 3	2	1	1	2	1
Prototipo 4	1	1	2	1	2
Prototipo 5	2	2	1	1	1
Prototipo 6	1	1	1	2	2
Prototipo 7	2	2	1	1	1

Tabla 3. Matriz vecinosClase

La matriz vecinos la necesitaremos más adelante, de momento nos centraremos en la matriz vecinosClase y la modificaremos para que el valor de las celdas de cada columna sea la moda de las anteriores y de ella misma. Lo que conseguimos es una matriz donde cada fila es la clase que predeciríamos para ese prototipo utilizando el algoritmo k-NN con todos los valores de k hasta k_{max}. Esta matriz que obtendremos se llamará clasePredK y se muestra en la Tabla 4. Es necesario comentar que en caso de empate con la moda, como ocurre por ejemplo en el Prototipo 3 con k = 4, donde tenemos dos clases 1 y dos clases 2 siempre elegimos la clase menor porque estamos trabajando con MATLAB y es la manera que tiene de resolver los casos de empate.

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	1	1	1	1	1
Prototipo 2	1	1	1	1	1
Prototipo 3	2	1	1	1	1
Prototipo 4	1	1	1	1	1
Prototipo 5	2	2	2	1	1
Prototipo 6	1	1	1	1	1
Prototipo 7	2	2	2	1	1

Tabla 4. Matriz clasePredK

Cada columna de la matriz clasePredK es la clase predicha para cada uno de los prototipos utilizando el algoritmo k-NN con cada valor de k. Nos interesa saber cuántas veces acertamos o fallamos con esa predicción, restaremos a cada una de las columnas de la matriz clasePredK la clase real de cada prototipo.

1
1
2
1
2
1
2

Tabla 5. Clase real de cada prototipo

-

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	1	1	1	1	1
Prototipo 2	1	1	1	1	1
Prototipo 3	2	1	1	1	1
Prototipo 4	1	1	1	1	1
Prototipo 5	2	2	2	1	1
Prototipo 6	1	1	1	1	1
Prototipo 7	2	2	2	1	1

Tabla 4. Matriz clasePredK

El resultado de esta resta se muestra en la tabla 6, es una matriz donde aquellas celdas cuyo valor sea 0 significa que hemos acertado con la predicción:

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	0	0	0	0	0
Prototipo 2	0	0	0	0	0
Prototipo 3	0	1	1	1	1
Prototipo 4	0	0	0	0	0
Prototipo 5	0	0	0	1	1
Prototipo 6	0	0	0	0	0
Prototipo 7	0	0	0	1	1

Tabla 6. Matriz aciertos

Por comodidad para los cálculos posteriores modificamos la matriz aciertos para que los aciertos pasen a ser unos y los errores ceros. Se muestra la matriz modificada en la tabla 7.

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	1	1	1	1	1
Prototipo 2	1	1	1	1	1
Prototipo 3	1	0	0	0	0
Prototipo 4	1	1	1	1	1
Prototipo 5	1	1	1	0	0
Prototipo 6	1	1	1	1	1
Prototipo 7	1	1	1	0	0

Tabla 7. Matriz aciertos modificada

Recuperamos ahora la matriz de vecinos que habíamos calculado al comienzo. A partir de esta matriz obtendremos para cada prototipo aquellos prototipos para los cuales es uno de sus 3 vecinos más cercanos, es decir, obtendremos el vecindario de cada prototipo. La

manera es muy sencilla, buscamos el prototipo para el que estamos calculando su vecindario en la matriz vecinos, en todas las filas en que se encuentre serán los prototipos de su vecindario. Como no todos los vecindarios van a ser del mismo tamaño, no podemos utilizar una matriz para guardar esta información, utilizaremos otra estructura de datos distinta en la que cada elemento apunta a un vector.

Mostramos como ejemplo el cálculo del vecindario del prototipo 3. En la matriz vecinos el prototipo 3 se encuentra en las filas 1, 4, 5 y 7, por tanto estos prototipos junto a él mismo conformarán su vecindario. Se muestran todos los vecindarios en la tabla 8.

Prototipo 1	4	3	2
Prototipo 2	6	4	1
Prototipo 3	7	1	4
Prototipo 4	1	2	3
Prototipo 5	7	3	6
Prototipo 6	2	1	4
Prototipo 7	3	5	1

Tabla 2. Matriz vecinos

Prototipo 1	2	3	4	6	7
Prototipo 2	1	4	6		
Prototipo 3	1	4	5	7	
Prototipo 4	1	2	3	6	
Prototipo 5	7				
Prototipo 6	2	5			
Prototipo 7	3	5			

Tabla 8. Vecindario de cada prototipo

Con esta estructura y la matriz aciertos ya podemos calcular el accuracy local. Debemos comprobar para cada vecindario, el acierto del k-NN con cada uno de los prototipos que lo conforman para todos los valores de k.

Mostraremos de ejemplo el cálculo del accuracy local para el vecindario del prototipo 3. Como ya hemos dicho, su vecindario está conformado por los prototipos 1, 4, 5, 7 y él mismo, por lo que tendremos que fijarnos sólo en esos prototipos.

	k = 1	**k = 2**	**k = 3**	**k = 4**	**k = 5**
Prototipo 1	1	1	1	1	1
Prototipo 2	1	1	1	1	1
Prototipo 3	1	0	0	0	0
Prototipo 4	1	1	1	1	1
Prototipo 5	1	1	1	0	0
Prototipo 6	1	1	1	1	1
Prototipo 7	1	1	1	0	0

Tabla 9. Matriz aciertos con el vecindario del prototipo 3 resaltado

Miramos para los 5 prototipos del vecindario cuántos clasificamos correctamente mediante k-NN con valor de k = 1, en este caso clasificamos los 5 correctamente, por tanto, el accuracy local del prototipo 3 con valor de k = 1 es 5/5 = 100%. Con valores de k = 2 y k = 3 el prototipo 3 se clasifica de manera errónea, el resto bien, el accuracy local será 4/5 = 80%. Para valores de k = 4 y k = 5 clasificamos correctamente sólo 2 prototipos, el accuracy será 2/5 = 40%.

Haciendo los mismo cálculos para todos los vecindarios obtendremos la matriz acc_{local} que es la segunda medida que necesitamos para calcular el accuracy total, a partir del cual elegiremos el valor óptimo de k local asociado a cada prototipo.

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	100	83.33	83.33	66.67	66.67
Prototipo 2	100	100	100	100	100
Prototipo 3	100	80	80	40	40
Prototipo 4	100	80	80	80	80
Prototipo 5	100	100	100	0	0
Prototipo 6	100	100	100	66.67	66.67
Prototipo 7	100	66.67	66.67	0	0

Tabla 10. Matriz acc_{local}

3.2.3 Accuracy total o eval

Esta medida es muy simple de calcular, sumamos el accuracy global a todos los prototipos del accuracy local, es decir, a cada fila de la matriz acc_{local} le sumamos el vector acc_{global} anteriormente calculado. El resultado se muestra en la tabla 11.

	k = 1	k = 2	k = 3	k = 4	k = 5
Prototipo 1	200	169,04	169,04	123,81	123,81
Prototipo 2	200	185,71	185,71	157,14	157,14
Prototipo 3	200	165,71	165,71	97,14	97,14
Prototipo 4	200	165,71	165,71	137,14	137,14
Prototipo 5	200	185,71	185,71	57,14	57,14
Prototipo 6	200	185,71	185,71	123,81	123,81
Prototipo 7	200	152,38	152,38	57,14	57,14

Tabla 11. Matriz eval

Aquel valor de k para el que el accuracy total sea mayor será el k local asociado a cada prototipo. En caso de empate elegimos aquel cuyo valor de k sea menor. En este caso para todos los prototipos nos sale que el valor óptimo de k es 1. Es normal, ya que hemos escogido un conjunto de ejemplos muy pequeño con sólo 2 clases que además están claramente separadas.

Una vez conocido el valor de k local óptimo de cada prototipo, guardamos ese valor junto a cada prototipo. En la tabla 12 se muestra cómo queda modificada la matriz con el conjunto de train.

x1	x2	y	k
0,3	0,6	1	1
0,2	0,8	1	1
0,5	0,5	2	1
0,2	0,6	1	1
0,7	0,8	2	1
0,3	0,9	1	1
0,7	0,5	2	1

Tabla 12. Conjunto de train con la k local

3.2.4 Clasificar nuevos ejemplos.

La parte más simple y sencilla, tan sólo hay que calcular para cada nuevo ejemplo su prototipo más cercano, mirar la k local asociada que tenga y clasificar el ejemplo teniendo en cuenta tantos vecinos como indique la k. Se muestra en la tabla 13 un conjunto de test

que utilizaremos como ejemplo. En la figura 3 se muestran mediante asteriscos los ejemplos del conjunto de test junto a los ejemplos del conjunto de train.

x1	x2	y
0,3	0,4	1
0,9	0,5	2
0,4	0,7	1

Tabla 13. Conjunto de test

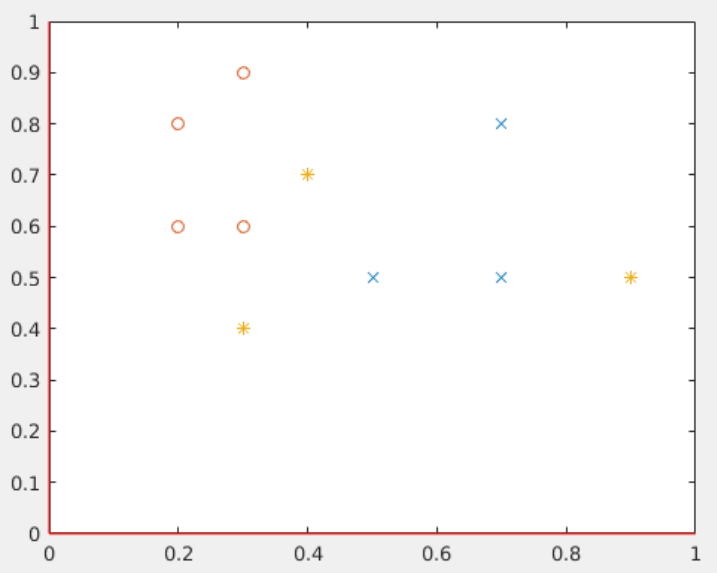


Figura 3. Gráfica con los conjuntos train y test

Los prototipos más cercanos a cada uno son el 1, el 7 y el 1 respectivamente. Al tener los 3 como k local asociada el 1, su propia clase será la clase con la que clasifiquemos a los ejemplos. Por tanto la clase predicha para cada uno de ellos será 1, 2 y 1. Acertaríamos al 100%.

4. Marco Experimental

El objetivo de este trabajo no es sólo el de implementar una variante del método k-NN estándar, sino también estudiar si ésta nueva variante es mejor que el método k-NN estándar entre otros. En esta sección explicaremos cuáles son los otros métodos contra los que compararemos el método k-NN local y los datasets elegidos sobre los que se ejecutarán. También comentaremos las medidas de rendimiento que calcularemos al ejecutar cada método y que necesitaremos para poder compararlos, así como los test estadísticos que utilizaremos para realizar la comparativa.

4.1 Métodos a comparar

En este proyecto compararemos el rendimiento de 5 métodos distintos:

- **1-NN**: el método más sencillo del algoritmo k-NN donde cada ejemplo es clasificado con la clase de su vecino más cercano del conjunto de train.

- **3-NN**: quizás el método más utilizado para el algoritmo k-NN, tendremos en cuenta los 3 vecinos más cercanos del conjunto de train para decidir la clase del ejemplo que estamos clasificando.

- **k-NN global**: en este método el valor de k asociado a cada ejemplo del conjunto de train vendrá dado únicamente por el accuracy global que calculábamos en la primera parte del método k-NN local. El valor de k desde 1 hasta k_{max} que obtenga mejor accuracy será el que vaya asociado a todos los prototipos, con lo cual clasificaremos los nuevos ejemplos con el algoritmo k-NN con el mismo valor de k para todos.

- **k-NN sólo local**: en este método el valor de k asociado a cada ejemplo del conjunto de train vendrá dado por el accuracy local que obteníamos en el método k-NN local. Éste valor de k podrá ser distinto para cada prototipo, clasificaremos cada nuevo ejemplo con el valor de k asociado que tenga su prototipo más cercano.

- **k-NN local**: el método que hemos explicado e implementado en la sección 3 y que queremos comprobar si mejora a los otros 4.

4.2 Datasets

Los datasets que utilizaremos para hacer las pruebas han sido sacados del repositorio de datasets de la página web keel.es [2]. A cada dataset se le ha hecho validación cruzada de 5 particiones, es decir, cada una de las 5 particiones será un conjunto de test y las otras 4 serán el conjunto de train con el que clasificar a esa partición. Hemos seleccionado 2 tipos de datasets distintos, por una parte unos datasets normales con número variable de clases y otros imbalanceados que tienen sólo 2 clases.

Dentro de los datasets normales hay 2 conjuntos distintos. Un conjunto, al que llamaremos “small” (tabla 14), está formado por 41 datasets con un número de ejemplos desde 150 hasta 3196, con un rango de número de atributos desde 2 hasta 90 y con un máximo de hasta 15

clases. El otro conjunto está formado por 20 datasets con un número de ejemplos desde 4174 hasta 58000, con un rango de número de atributos desde 2 hasta 85 y con un máximo de hasta 28 clases. A este conjunto lo llamaremos “big” (tabla 15), ya que en él están los datasets más grandes.

Como ya hemos mencionado los datasets imbalanceados sólo tienen 2 clases, hemos dividido estos datasets en 2 conjuntos en función del ratio de imbalanceo. El ratio de imbalanceo no es más que la proporción de ejemplos que hay de más de una clase que de la otra. Aquellos cuyo ratio de imbalanceo es menor que 9 están en el conjunto “imb1” (tabla 16), son un total de 22 datasets con un rango de ejemplos desde 150 hasta 5742, con un rango de número de atributos desde 3 hasta 19. Aquellos datasets con un ratio mayor que 9 conforman el conjunto “imb2” (tabla 17), contiene 73 datasets con un rango de ejemplos desde 168 hasta 4174, con un rango de número de atributos desde 6 hasta 41.

En el caso de los datasets imbalanceados, dado que vamos a tener muchos más ejemplos de una clase que de otra lo que más nos interesa es comprobar cómo de bien estamos clasificando la clase con menos atributos. Por tanto, es necesario explicar una modificación que realizamos al método k-NN local anteriormente expuesto cuando estamos ejecutando el algoritmo sobre los datasets imbalanceados.

Como decíamos, para obtener la k local de cada prototipo calculábamos una medida llamada eval que era la suma del accuracy global y del accuracy local obtenido. El accuracy global lo calculábamos realizando validación cruzada de 10 particiones sobre el conjunto de train y ejecutando k-NN con todos los valores de k para cada una de esas 10 particiones. Así obteníamos 10 medidas de accuracy y la media de todas ellas era el accuracy global. La modificación consiste en que ahora no nos interesa tanto el acccuracy obtenido en cada una de esas 10 particiones ya que puede hacer que una clase solape a otra, sino la media geométrica que nos indica cómo de bien estamos clasificando ambas.

En resumen, cuando ejecutamos el algoritmo para los datasets imbalanceados, el accuracy global será la media geométrica obtenida haciendo validación cruzada de 10 particiones sobre el conjunto de train y cuando ejecutemos el algoritmo con los datasets normales el accuracy global será como habíamos expuesto anteriormente. El resto de la ejecución no se ve modificada.

A continuación mostramos los datasets de los 4 conjuntos mencionados, junto con su número de ejemplos, atributos y clases:

Dataset	Ejemplos	Atributos	Clases
australian	690	14	2
balance	625	4	3
bands	365	19	2
breast	277	9	2
bupa	345	6	2
car	1728	6	4
chess	3196	36	2
cleveland	297	13	5
contraceptive	1473	9	3
crx	653	15	2
dermatology	358	34	6
ecoly	336	7	8
flare	1066	11	6
german	1000	20	2
glass	214	9	7
haberman	306	3	2
hayes-roth	160	4	3
heart	270	13	2
housevotes	232	16	2
ionosphere	351	33	2
iris	150	4	3
led7digit	500	7	10
mammographic	830	5	2
monk-2	432	6	2
movement-libras	360	90	15
newthyroid	215	5	3
pima	768	8	2
saheart	462	9	2
segment	2310	19	7
sonar	208	60	2
spectfheart	267	44	2
splice	3190	60	3
tic-tac-toe	958	9	2
titanic	2201	3	2
vehicle	846	18	4
vowel	990	13	11
wdbc	569	30	2
wine	178	13	3
winequality-red	1599	11	11
wisconsin	683	9	2
yeast	1484	8	10

Tabla 14. Conjunto de datasets “small”

Dataset	Ejemplos	Atributos	Clases
abalone	4174	8	28
banana	5300	2	2
coil2000	9822	85	2
kr-vs-k	28056	6	18
letter	20000	16	26
magic	19020	10	2
marketing	6876	13	9
mushroom	5644	22	2
nursery	12960	8	5
page-blocks	5472	10	5
penbased	10992	16	10
phoneme	5404	5	2
ring	7400	20	2
satimage	6435	36	7
shuttle	57999	9	7
spambase	4597	57	2
texture	5500	40	11
thyroid	7200	21	3
twonorm	7400	20	2
winequality-white	4898	11	11

Tabla 15. Conjunto de datasets “big”

Dataset	Ejemplos	Atributos	Clases
ecoli-0vs1	220	7	2
ecoli1	336	7	2
ecoli2	336	7	2
ecoli3	336	7	2
glass-0123vs456	214	9	2
glass0	214	9	2
glass1	214	9	2
glass6	214	9	2
haberman	306	3	2
iris0	150	4	2
new-thyroid1	215	5	2
new-thyroid2	215	5	2
page-blocks0	5472	10	2
pima	768	8	2
segment0	2308	19	2
vehicle0	846	18	2
vehicle1	846	18	2
vehicle2	846	18	2
vehicle3	846	18	2
wisconsin	683	9	2
yeast1	1484	8	2
yeast3	1484	8	2

Tabla 16. Conjunto de datasets “imb1”

Dataset	Ejemplos	Atributos	Clases
abalone-17vs7-8-9-10	2338	8	2
abalone-19vs10-11-12-13	1622	8	2
abalone-20vs8-9-10	1916	8	2
abalone-21vs8	581	8	2
abalone-3vs11	502	8	2
abalone19	4174	8	2
abalone-9vs18	731	8	2
car-good	1728	6	2
car-vgood	1728	6	2
cleveland-0vs4	173	13	2
dermatology6	358	34	2
ecoli-0-1-3-7vs2-6	281	7	2
ecoli-0-1-4-6vs5	280	6	2
ecoli-0-1-4-7vs2-3-5-6	336	7	2
ecoli-0-1-4-7vs5-6	332	6	2
ecoli-0-1vs2-3-5	244	7	2
ecoli-0-1vs5	240	6	2
ecoli-0-2-3-4vs5	202	7	2
ecoli-0-2-6-7vs3-5	224	7	2
ecoli-0-3-4-6vs5	205	7	2
ecoli-0-3-4-7vs5-6	257	7	2
ecoli-0-3-4vs5	200	7	2
ecoli-0-4-6vs5	203	6	2
ecoli-0-6-7vs3-5	222	7	2
ecoli-0-6-7vs5	220	6	2
ecoli4	336	7	2
flare-F	1066	11	2
glass-0-1-4-6vs2	205	9	2
glass-0-1-5vs2	172	9	2
glass-0-1-6vs2	192	9	2
glass-0-1-6vs5	184	9	2
glass2	214	9	2
glass4	214	9	2
glass5	214	9	2
kddcup-bufferOverflow-vs-back	2233	41	2
kddcup-guessPasswd-vs-satan	1642	41	2
kddcup-land-vs-portsweep	1061	41	2
kddcup-land-vs-satan	1610	41	2
kddcup-rootkitImap-vs-back	2225	41	2
kr-vs-kOne-vs-fifteen	2244	6	2
kr-vs-kThree-vs-eleven	2935	6	2
kr-vs-kZeroOne-vs-draw	2901	6	2
kr-vs-kZero-vs-eight	1460	6	2
kr-vs-kZero-vs-fifteen	2193	6	2
led7digit-0-2-4-5-6-7-8-9vs1	443	7	2
page-blocks-1-3vs4	472	10	2
poker-8-9vs5	2075	10	2
poker-8-9vs6	1485	10	2
poker-8vs6	1477	10	2
poker-9vs7	244	10	2
shuttle-2vs5	3316	9	2
shuttle-6vs2-3	230	9	2
shuttle-c0-vs-c4	1829	9	2
vowel0	988	13	2

winequality-red-3vs5	691	11	2
winequality-red-4	1599	11	2
winequality-red-8vs6	656	11	2
winequality-red-8vs6-7	855	11	2
winequality-white-3-9vs5	1482	11	2
winequality-white-3vs7	900	11	2
winequality-white-9vs4	168	11	2
yeast-0-2-5-6vs3-7-8-9	1004	8	2
yeast-0-2-5-7-9vs3-6-8	1004	8	2
yeast-0-3-5-9vs7-8	506	8	2
yeast-0-5-6-7-9vs4	528	8	2
yeast-1-2-8-9vs7	947	8	2
yeast-1-4-5-8vs7	693	8	2
yeast-1vs7	459	7	2
yeast-2vs4	514	8	2
yeast-2vs8	482	8	2
yeast4	1484	8	2
yeast5	1484	8	2
yeast6	1484	8	2

Tabla 17. Conjunto de datasets “imb2”

4.3 Medidas y test estadísticos

4.3.1 Medidas

Las pruebas se realizarán de la siguiente manera, ejecutaremos para cada dataset los 5 métodos de k-NN a comparar en el siguiente orden: 1-NN, 3-NN, k-NN global, k-NN sólo local y k-NN local. Dado que cada datasets está dividido en 5 particiones, repetiremos este proceso con cada partición del dataset. De cada uno de los métodos calcularemos 5 medidas: accuracy, tiempo de ejecución, media geométrica, accuracy medio por clases y medida k de Cohen.

El tiempo lo calcularemos para comprobar no sólo si el nuevo método es más preciso que los otros, si no si es eficiente en tiempo de ejecución. No es lo mismo que el k-NN local mejore a los otros con un tiempo de ejecución más o menos similar a que los mejore pero tarde 10 veces más en ejecutarse.

El accuracy, media geométrica, accuracy medio por clases y medida k de Cohen son medidas obtenidas a partir de la matriz de confusión como ya hemos explicado en el apartado 2. Accuracy y medida k de Cohen nos dará un indicador de cómo de preciso es nuestro clasificador. El accuracy medio por clases y sobretodo la media geométrica nos servirán para comprobar si todas las clases se clasifican bien ya que es posible que con el accuracy una clase mal clasificada quede solapada por otras que tengan un acierto muy alto.

Cada dataset hemos dicho que ha sido dividido en 5 particiones, por tanto nuestro programa ejecutará los 5 métodos distintos de k-NN mencionados en las 5 particiones de cada dataset obteniendo así el accuracy, accuracy medio por clases, media geométrica, medida k de Cohen y tiempo en cada una de ellas. Haciendo la media de todas ellas obtendremos un

único valor medio para cada datasets con cada una de las medidas. Este valor medio de cada medida será con el que utilicemos en los test.

4.3.2 Test estadísticos

Hemos escogido 2 test distintos para comparar los 5 métodos expuestos anteriormente y comprobar cuál de ellos es el mejor, si es que se puede decir que uno de ellos sea mejor que los otros:

- **Test de rangos alineados de Friedman**: Este test lo hemos elegido principalmente porque permite comparar varios métodos a la vez. Trabaja a partir de los resultados obtenidos en cada datasets. Se debe realizar una vez por cada medida. En función de los resultados obtenidos por cada método va calculando una serie de rangos, al acabar de ejecutarse el método que tenga un número de rango menor será supuestamente el mejor. Tras eso realizaremos otro test llamado test de Hommel que calcula una medida llamada p_{valor} para cada método con el método que ha obtenido mejor rango. Éstos p valores son los que realmente nos indican si hay diferencias estadísticas entre el método de menor rango con los demás. Si el p_{valor} es menor o igual a 0.05 se puede afirmar que el método de menor rango es estadísticamente mejor que el otro.

- **Test de Wilcoxon**: Este test sólo permite comparar un método frente a otro. Nos será útil para aquellas veces en que el test de Hommel realizado tras el de rangos alineados de Friedman no consiga obtener diferencias estadísticas entre los métodos. Se le deben pasar los resultados obtenidos por ambos métodos en todos los datasets para una medida concreta. Su ejecución es muy sencilla, en cada dataset comprueba quién gana, es decir, quién obtiene mejor resultado y por cuánta diferencia respecto al otro método. Al final de su ejecución el método con rango mayor será supuestamente el mejor. También este test calcula el p_{valor} que nos dirá si hay diferencias significativas entre ambos. Si es menor o igual que 0.05 se podrá afirmar que el de rango mayor es mejor.

5. Estudio Experimental

Hemos ejecutado los 5 métodos anteriormente explicados (1-NN, 3-NN, k-NN global, k-NN sólo local y k-NN local) sobre los 4 conjuntos de datasets mencionados (small, big, imb1 e imb2) y hemos realizado los test para comprobar si el método k-NN local presentado es mejor que los otros 4.

5.1 Resultados

Mostraremos a continuación las tablas con los resultados obtenidos. De los conjuntos imbalanceados lo que más nos interesa conocer es el accuracy medio por clases y la media geométrica, ya que al haber sólo 2 clases, con una en una proporción mucho mayor que la otra, el accuracy y la medida k de Cohen podrían dar resultados engañosos. Por tanto, para los datasets imbalanceados mostraremos sólo el accuracy medio por clases y la media geométrica y para los datasets normales mostraremos también el accuracy y la medida k de Cohen.

5.1.1Conjunto “small”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	81,0137	84,6360	85,6589	85,3607	**85,6599**
balance	79,8315	84,3274	**90,0893**	87,3587	89,9293
bands	**71,2329**	69,8630	70,4110	68,7671	67,1233
breast	68,1404	69,6459	**73,2695**	72,1659	72,8804
bupa	63,4783	63,7681	**64,3478**	60,8696	**64,3478**
car	87,7900	92,5921	94,2136	90,0474	**94,9079**
chess	90,7060	**95,4632**	95,2759	92,3958	94,9936
cleveland	51,8480	**56,8819**	55,5488	55,8876	55,5373
contraceptive	44,8110	46,7128	**49,0787**	47,0547	48,9505
crx	80,3965	83,7706	**86,6726**	83,1517	84,8358
dermatology	94,6813	**96,6416**	95,5384	94,9590	94,4114
ecoly	80,6274	82,5568	82,2491	**83,6683**	82,8378
flare	62,9872	62,9319	**71,3533**	69,0022	69,2796
german	68,2000	69,0000	**72,8000**	70,4000	72,3000
glass	71,4607	70,1295	71,5026	**71,9557**	71,0480
haberman	69,9260	72,8609	**75,8170**	74,8281	73,8498
hayes-roth	40,4264	32,3760	**73,1934**	71,8438	66,8402
heart	75,5556	77,0370	82,9630	78,5185	**83,3333**
housevotes	**92,3092**	91,0141	90,5885	**92,3092**	90,5885
ionosphere	87,1791	86,0362	85,7505	**89,4567**	88,3179
iris	**95,3333**	**95,3333**	**95,3333**	**95,3333**	**95,3333**
led7digit	30,9721	37,5801	**71,8131**	69,0455	70,7098
mammographic	75,2998	81,2131	**81,6871**	79,2824	80,9663
monk-2	77,2933	**95,3619**	**95,3619**	84,4898	**95,3619**
movement-libras	80,8543	77,9211	**84,7986**	82,3283	82,9319
newthyroid	**96,7442**	94,4186	**96,7442**	**96,7442**	**96,7442**
pima	70,1749	74,4691	73,0430	**75,2559**	74,8638
saheart	66,6667	68,1814	**72,7302**	67,0968	70,7691
segment	96,7532	95,7576	**97,0563**	96,6667	96,6667
sonar	86,5357	84,1769	**87,5113**	85,1182	84,6647
spectfheart	70,0210	71,8798	**80,5311**	74,5353	77,9385
splice	75,2360	77,2121	87,0536	81,6641	**87,3053**
tic-tac-toe	81,8296	86,0055	86,3197	86,1108	**86,9480**
titanic	71,8493	71,4700	**79,2223**	77,4348	78,3323
vehicle	70,4569	71,0494	71,0411	**71,1559**	70,4590
vowel	**99,2929**	97,4747	**99,2929**	98,9899	99,1919
wdbc	95,2520	97,0096	**97,0127**	95,6029	**97,0127**
wine	95,4576	96,6173	97,1887	95,4576	**97,2037**
winequality-red	53,4699	52,6552	58,9159	**63,9795**	62,5516
wisconsin	95,9016	96,9257	96,7797	96,3439	**96,9278**
yeast	52,2857	55,3221	59,9706	58,2131	**60,3126**
Media	75,6166	77,2263	**81,3592**	79,7768	80,8577

Tabla 18. Accuracy para el conjunto de datasets “small”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	80,8007	84,4553	**85,6736**	85,1709	85,5427
balance	57,7475	61,0002	**65,1664**	63,1922	65,0514
bands	**68,4541**	66,9082	67,1900	65,5797	63,2045
breast	59,6883	59,7249	59,0260	**60,7394**	59,4333
bupa	62,4310	**63,0603**	61,1897	60,4655	62,8017
car	69,5667	**79,5358**	76,1658	70,2890	79,1987
chess	90,4887	**95,3446**	95,1652	92,2046	94,8591
cleveland	**33,6380**	29,6146	26,9922	31,7435	26,3435
contraceptive	43,5294	44,2084	**47,4555**	44,4645	46,6754
crx	79,9357	83,6797	**87,0588**	83,1729	84,9130
dermatology	93,9007	**96,6448**	94,4815	94,4259	93,9411
ecoly	70,8189	72,8427	71,5411	**73,0238**	72,0490
flare	48,8606	47,0002	**57,9671**	55,3378	55,1146
german	**61,7619**	59,8571	59,2381	60,2857	59,8333
glass	66,9881	65,7202	**67,3016**	63,2024	66,4544
haberman	59,3815	58,5989	58,2190	**64,2892**	57,6912
hayes-roth	42,7961	31,0623	**69,8901**	69,3040	65,0733
heart	75,5833	76,5833	82,0833	78,5000	**82,7500**
housevotes	92,5664	91,2902	90,8902	**92,6209**	90,8902
ionosphere	82,9966	81,0564	81,1744	**86,0034**	84,4188
iris	**95,3333**	**95,3333**	**95,3333**	**95,3333**	**95,3333**
led7digit	32,8772	39,7328	**72,9264**	69,4946	71,5742
mammographic	75,2552	81,1442	81,7242	79,2398	80,9601
monk-2	76,6096	**95,4315**	**95,4315**	84,2316	**95,4315**
movement-libras	80,8444	77,9556	**84,7778**	82,3556	82,9556
newthyroid	**95,9365**	90,6984	**95,9365**	**95,9365**	**95,9365**
pima	66,3484	70,3438	66,5926	**70,5039**	69,9588
saheart	62,3029	62,2920	**64,1551**	60,4242	63,0926
segment	96,7532	95,7576	**97,0563**	96,6667	96,6667
sonar	86,3481	83,7501	**87,6161**	85,1610	84,8243
spectfheart	59,5580	56,6949	56,0989	**58,3671**	55,8019
splice	75,8695	78,1665	86,5101	82,6560	**87,2319**
tic-tac-toe	75,2557	81,5031	81,7431	81,2977	**82,5075**
titanic	61,9950	61,9629	**69,3831**	67,6563	68,6220
vehicle	70,6471	**71,3076**	71,3033	71,3928	70,7252
vowel	**99,2929**	97,4747	**99,2929**	98,9899	99,1919
wdbc	94,7759	96,3639	**96,5603**	94,9584	**96,5603**
wine	96,2222	97,1746	**97,6508**	96,2222	97,4603
winequality-red	28,1316	28,3267	27,6915	**32,4780**	29,6122
wisconsin	95,3985	**96,7684**	96,3650	95,9364	96,5778
yeast	53,8878	54,9563	49,5246	57,2443	**58,0538**
Media	71,2580	72,2275	**75,0620**	74,4039	75,0077

Tabla 19. Accuracy medio por clases para el conjunto de datasets “small”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	80,7398	84,3537	**85,5002**	85,0770	85,3809
balance	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
bands	**67,2315**	65,9263	65,7845	64,3337	61,1678
breast	**55,8558**	54,6979	47,8393	54,0639	49,5343
bupa	62,0234	**62,8301**	57,8146	60,2704	61,9339
car	66,8764	**78,2624**	72,2270	67,2322	76,1089
chess	90,3474	**95,3043**	95,1287	92,0902	94,8010
cleveland	5,6119	0,0000	0,0000	**6,3119**	0,0000
contraceptive	43,0302	42,7712	**46,7837**	43,1176	45,6624
crx	79,7325	83,6090	**86,9244**	83,1332	84,8301
dermatology	93,3590	**96,4098**	93,9400	93,7515	93,1135
ecoly	44,9186	**49,2556**	47,6954	48,5040	48,6991
flare	38,4180	35,8498	**44,7871**	40,9933	32,5809
german	**59,5059**	55,0112	48,3244	54,3195	50,7690
glass	**52,0780**	38,1113	39,4374	26,1672	27,8488
haberman	54,6487	49,9151	40,0279	**59,9989**	45,9065
hayes-roth	40,6317	29,9254	66,1284	66,0422	**62,1836**
heart	75,1173	76,2019	81,5006	78,3092	**82,4762**
housevotes	92,4666	91,1815	90,7601	**92,4860**	90,7601
ionosphere	81,3989	79,0298	79,2821	**85,0361**	83,2164
iris	**95,1115**	**95,1115**	**95,1115**	**95,1115**	**95,1115**
led7digit	0,0000	0,0000	**70,9906**	67,7344	69,6880
mammographic	75,1147	81,0610	**81,5363**	79,0604	80,8005
monk-2	75,5844	**95,4120**	**95,4120**	84,0039	95,4092
movement-libras	76,6301	46,6008	**82,7074**	80,0092	80,5218
newthyroid	**95,7946**	90,0873	**95,7946**	**95,7946**	**95,7946**
pima	65,0525	**68,9663**	62,8925	68,6022	67,9896
saheart	**60,5987**	59,0358	57,5462	55,9114	57,8270
segment	96,6483	95,6311	**96,9774**	96,5699	96,5699
sonar	86,2379	83,4395	**87,4588**	84,9821	84,5639
spectfheart	**55,2210**	49,4016	37,1673	50,0299	36,1651
splice	75,8106	78,0987	86,3739	82,5816	**87,1650**
tic-tac-toe	71,9347	80,1089	80,2623	79,5754	**81,1095**
titanic	51,0486	53,0832	**62,4846**	59,8902	61,9680
vehicle	67,6257	**66,8007**	66,2852	66,8000	65,8856
vowel	**99,2760**	97,3929	**99,2760**	98,9616	99,1672
wdbc	94,7521	96,3142	**96,5357**	94,9202	**96,5357**
wine	96,0228	97,0539	**97,5933**	96,0228	97,4068
winequality-red	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
wisconsin	95,3730	**96,7650**	96,3518	95,9230	96,5656
yeast	**29,9814**	0,0000	0,0000	9,5769	0,0000
Media	64,5807	63,3905	66,7962	**66,9097**	66,4200

Tabla 20. Media geométrica para el conjunto de datasets “small”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	0,6157	0,6891	**0,7104**	0,7035	0,7097
balance	0,6438	0,7158	**0,8162**	0,7680	0,8132
bands	**0,3751**	0,3472	0,3529	0,3196	0,2747
breast	0,2075	0,2159	0,2183	**0,2452**	0,2249
bupa	0,2492	0,2598	0,2343	0,2065	**0,2599**
car	0,7252	0,8356	0,8705	0,7756	**0,8861**
chess	0,8130	**0,9089**	0,9051	0,8470	0,8994
cleveland	0,2705	0,2825	0,2467	**0,2929**	0,2569
contraceptive	0,1507	0,1624	**0,2132**	0,1716	0,2033
crx	0,6022	0,6728	**0,7338**	0,6611	0,6954
dermatology	0,9333	**0,9579**	0,9440	0,9368	0,9300
ecoly	0,7335	0,7575	0,7519	**0,7726**	0,7605
flare	0,5177	0,5104	**0,6263**	0,5949	0,5982
german	**0,2374**	0,2095	0,2218	0,2249	0,2289
glass	0,6125	0,5869	0,6097	**0,6128**	0,6016
haberman	0,1951	0,1979	0,2056	**0,3071**	0,1854
hayes-roth	0,0314	-0,1165	**0,5657**	0,5463	0,4650
heart	0,5077	0,5329	0,6498	0,5667	**0,6595**
housevotes	0,8464	0,8206	0,8122	**0,8465**	0,8122
ionosphere	0,7008	0,6710	0,6659	**0,7577**	0,7300
iris	**0,9300**	**0,9300**	**0,9300**	**0,9300**	**0,9300**
led7digit	0,2385	0,3103	**0,6866**	0,6551	0,6740
mammographic	0,5053	0,6236	**0,6338**	0,5851	0,6191
monk-2	0,5389	**0,9071**	**0,9071**	0,6876	**0,9071**
movement-libras	0,7949	0,7634	**0,8371**	0,8107	0,8171
newthyroid	**0,9305**	0,8750	**0,9305**	**0,9305**	**0,9305**
pima	0,3325	0,4203	0,3595	**0,4285**	0,4191
saheart	0,2503	0,2582	**0,3203**	0,2213	0,2898
segment	0,9621	0,9505	**0,9657**	0,9611	0,9611
sonar	0,7291	0,6797	**0,7501**	0,7019	0,6936
spectfheart	0,1705	0,1389	0,1701	**0,1774**	0,1479
splice	0,6095	0,6426	0,7912	0,7106	**0,7964**
tic-tac-toe	0,5558	0,6704	0,6768	0,6697	**0,6922**
titanic	0,2609	0,2602	**0,4470**	0,3981	0,4283
vehicle	0,6059	0,6140	0,6139	**0,6153**	0,6061
vowel	**0,9922**	0,9722	**0,9922**	0,9889	0,9911
wdbc	0,8983	0,9353	**0,9358**	0,9054	**0,9358**
wine	0,9314	0,9488	0,9574	0,9314	**0,9576**
winequality-red	0,2719	0,2480	0,3300	**0,4236**	0,3891
wisconsin	0,9098	**0,9326**	0,9291	0,9196	0,9324
yeast	0,3857	0,4207	0,4767	0,4573	**0,4821**
Media	0,5554	0,5785	**0,6340**	0,6163	0,6292

Tabla 21. Medida k de Cohen para el conjunto de datasets “small”

Comprobamos que para el accuracy, accuracy medio por clases y la medida k de Cohen el método k-NN global es con el que se obtiene un mejor valor medio, sin embargo para la media geométrica el método k-NN sólo local es el que mejor media da. Entendibles estos resultados, mientras que el método k-NN global calcula la k en base a la precisión en todo el conjunto de train, el k-NN sólo local calcula la k respecto a la precisión por vecindario de cada prototipo. Por lo tanto con el k-NN sólo local en general todas las clases estarán clasificadas por igual, mientras que con el k-NN global estará mucho mejor clasificada la clase con más ejemplos con respecto al resto. En principio parece que el método local no va

a mejorar al resto y que el método global será el mejor, pero hasta que no hagamos los test no podremos afirmarlo.

5.1.2 Conjunto “big”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
abalone	21,3320	23,0599	**28,5608**	25,0925	27,9781
banana	87,8676	89,4714	**90,4343**	89,2830	89,8303
coil2000	90,5926	93,1888	**94,0338**	93,0869	93,8098
kr-vs-k	58,4802	62,1119	**78,9935**	68,8850	77,3186
letter	96,2496	95,9755	**96,2946**	96,0549	96,1401
magic	80,6992	82,9916	**83,8171**	82,7392	83,7856
marketing	28,4176	28,9269	33,8871	31,4580	**33,9737**
mushroom	**100,0000**	**100,0000**	**100,0000**	**100,0000**	**100,0000**
nursery	80,9406	93,4028	**93,4570**	86,4734	93,0171
page-blocks	95,8346	95,8891	95,7798	**96,1997**	96,1814
penbased	99,4180	99,3634	99,3816	**99,4362**	99,3907
phoneme	90,4328	89,9150	90,7103	90,6552	**90,9143**
ring	74,4311	71,3228	**80,9177**	76,6340	80,5799
satimage	91,0023	**91,7017**	91,3909	91,2356	91,2981
shuttle	**99,9379**	99,9155	**99,9379**	99,9293	99,9293
spambase	89,4893	89,5076	90,4229	**91,0331**	90,9225
texture	**99,1818**	98,9455	**99,1818**	99,1455	99,1273
thyroid	92,7226	**94,1530**	94,1113	93,5420	94,0559
twonorm	94,7162	96,5136	**97,5677**	95,8921	97,4596
winequality-white	49,4314	48,3447	**65,0125**	61,5189	61,1104
Media	81,0589	82,2350	**85,1946**	83,4147	84,8411

Tabla 22. Accuracy para el conjunto de datasets “big”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
abalone	12,9858	12,9590	13,2099	**13,6193**	13,2433
banana	87,7542	89,2591	**90,0647**	88,9501	89,4780
coil2000	**53,6075**	51,7093	50,0793	52,0554	50,3605
kr-vs-k	58,6971	60,1588	**72,5725**	66,1832	70,5635
letter	96,2296	95,9554	**96,2740**	96,0324	96,1174
magic	77,5746	79,3493	79,2401	79,0691	**79,5511**
marketing	24,8075	24,7307	27,2730	26,4285	**27,8752**
mushroom	**100,0000**	**100,0000**	**100,0000**	**100,0000**	**100,0000**
nursery	75,7128	**84,1268**	82,3612	79,8362	83,5984
page-blocks	**77,4544**	72,3025	72,0083	75,7541	73,7826
penbased	99,4146	99,3636	99,3803	**99,4306**	99,3875
phoneme	87,5539	86,7081	87,9347	87,6007	**88,1343**
ring	74,1852	71,0423	**80,7383**	76,4086	80,3957
satimage	89,7313	**90,2503**	89,6696	89,7942	89,5614
shuttle	90,2443	**91,7719**	90,2443	89,7588	89,7588
spambase	89,0421	88,8454	89,9663	**90,5753**	90,3883
texture	**99,1818**	98,9455	**99,1818**	99,1455	99,1273
thyroid	**63,3670**	58,2497	55,4018	61,3163	57,0631
twonorm	94,7161	96,5134	**97,5674**	95,8917	97,4593
winequality-white	28,2822	26,1467	**41,3530**	34,3430	33,3937
Media	74,0271	73,9194	**75,7260**	75,1097	75,4620

Tabla 23. Accuracy medio por clases para el conjunto de datasets “big”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
abalone	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
banana	87,7426	89,2347	**89,9930**	88,8881	89,4090
coil2000	**32,9091**	20,8215	1,8407	22,9880	8,7255
kr-vs-k	57,4370	58,7867	57,6702	**65,0080**	56,1854
letter	96,1930	95,9125	**96,2372**	95,9914	96,0780
magic	76,8505	**78,3855**	77,7175	78,0899	78,2578
marketing	22,1555	20,5118	19,7428	**22,2358**	21,7553
mushroom	**100,0000**	**100,0000**	**100,0000**	**100,0000**	**100,0000**
nursery	61,4868	69,3570	67,8028	66,0138	**69,3992**
page-blocks	**75,2937**	69,6852	69,2035	73,6487	71,4606
penbased	99,4132	99,3614	99,3784	**99,4290**	99,3854
phoneme	87,2513	86,3474	87,6518	87,2752	**87,8649**
ring	69,4682	64,5120	**78,4202**	72,6036	77,9420
satimage	89,3320	**89,8031**	89,1146	89,3255	88,9875
shuttle	57,6430	**73,4727**	57,6430	72,4619	72,4619
spambase	89,0073	88,7854	89,9305	**90,5438**	90,3467
texture	**99,1767**	98,9337	**99,1767**	99,1395	99,1209
thyroid	**55,0144**	45,0274	37,6459	50,9400	42,4102
twonorm	94,7142	96,5117	**97,5656**	95,8897	97,4575
winequality-white	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
Media	67,5544	67,2725	65,8367	**68,5236**	67,3624

Tabla 24. Media geométrica para el conjunto de datasets “big”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
abalone	0,1202	0,1282	**0,1769**	0,1527	0,1757
banana	0,7548	0,7867	**0,8056**	0,7824	0,7934
coil2000	**0,0782**	0,0528	0,0029	0,0623	0,0126
kr-vs-k	0,5372	0,5796	**0,7654**	0,6530	0,7467
letter	0,9610	0,9581	**0,9615**	0,9590	0,9599
magic	0,5658	0,6114	0,6224	0,6057	**0,6244**
marketing	0,1836	0,1873	0,2344	0,2137	**0,2378**
mushroom	**1,0000**	**1,0000**	**1,0000**	**1,0000**	**1,0000**
nursery	0,7213	0,9033	**0,9038**	0,8022	0,8975
page-blocks	0,7760	0,7644	0,7634	**0,7891**	0,7859
penbased	0,9935	0,9929	0,9931	**0,9937**	0,9932
phoneme	0,7648	0,7510	0,7718	0,7693	**0,7767**
ring	0,4860	0,4231	**0,6169**	0,5305	0,6101
satimage	0,8890	**0,8976**	0,8937	0,8919	0,8925
shuttle	**0,9983**	0,9976	**0,9983**	0,9980	0,9980
spambase	0,7802	0,7796	0,7995	**0,8122**	0,8096
texture	**0,9910**	0,9884	**0,9910**	0,9906	0,9904
thyroid	0,4220	0,4231	0,3725	**0,4316**	0,3981
twonorm	0,8943	0,9303	**0,9514**	0,9178	0,9492
winequality-white	0,2496	0,2256	**0,4800**	0,4150	0,4061
Media	0,6583	0,6691	**0,7052**	0,6885	0,7029

Tabla 25. Medida k de Cohen para el conjunto de datasets “big”

Obtenemos unos resultados muy similares al conjunto de datasets más pequeños. Para el accuracy, accuracy medio por clases y medida k de Cohen el mejor método en media es el global, con el método local que es el que mejor media tiene para la media geométrica. Hasta que no hagamos los test no podremos saber cuál es mejor o si hay un método mejor.

5.1.3 Conjunto “imb1”

Antes de mostrar los resultados para los datasets imbalanceados debemos explicar un par de puntos. En primer lugar, dado que estos datasets sólo tienen 2 clases, con una en mayor proporción que la otra (esta diferencia viene dada por el ratio de imbalanceo), quitaremos de la clase con más ejemplos tantos como haga falta para igualar el número de ejemplos de ambas clases. Obviamente quitamos ejemplos en el conjunto de train, para aprender una k en base a un conjunto que tiene el mismo número de ejemplos de ambas clases. En el conjunto de test no haremos modificaciones. Esta técnica se conoce como undersampling, quitamos ejemplos de manera aleatoria de la clase que tiene más, por tanto en cada ejecución obtendremos resultados distintos en cada ejecución. Esto nos lleva al segundo punto, para evitar este problema ejecutaremos todo el programa 5 veces y la media de todos los resultados obtenidos serán nuestros resultados finales.

Como ya hemos comentado para los datasets imbalanceados nos interesa lo bien que clasificamos ambas clases por tanto nos fijamos en las medidas accuracy medio por clases y media geométrica.

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
ecoli-0vs1	96,1946	**97,5905**	96,5873	97,5123	97,4609
ecoli1	86,4947	**89,5198**	88,1221	88,7367	88,5411
ecoli2	87,7412	90,1153	90,1900	**90,2499**	90,2257
ecoli3	85,1941	88,5037	**88,5399**	87,3350	88,5377
glass-0123vs456	**93,5511**	92,9674	92,8731	93,4511	92,6511
glass0	80,3399	79,8350	80,2463	**80,4680**	80,3399
glass1	**78,3265**	74,2468	73,6007	77,0913	73,0988
glass6	**89,0649**	88,3153	87,2342	88,3622	88,6667
haberman	**56,7717**	55,3077	55,2335	55,7884	56,3877
iris0	**100,0000**	**100,0000**	**100,0000**	**100,0000**	**100,0000**
new-thyroid1	97,7698	**98,5556**	97,8254	97,4286	97,8810
new-thyroid2	97,8254	**98,0952**	96,1111	97,5952	96,3968
page-blocks0	90,9925	91,0735	90,6673	**91,1813**	91,1582
pima	67,2509	70,6963	**72,0450**	71,3509	71,9874
segment0	98,7114	98,0040	**98,7366**	98,6103	98,6204
vehicle0	91,6334	91,5881	91,5333	**91,6754**	91,3785
vehicle1	66,7104	**71,4407**	70,5857	69,6522	71,3452
vehicle2	92,2073	91,7736	**92,2941**	92,0458	92,0615
vehicle3	71,8607	72,8045	72,6691	73,1568	**73,6563**
wisconsin	96,6686	97,3098	97,2523	**97,3966**	97,3573
yeast1	64,8702	68,4887	**70,8390**	69,1060	70,3844
yeast3	87,1022	89,4871	90,7113	89,0672	**90,9552**
Media	85,3310	86,1690	86,0862	86,2391	**86,3224**

Tabla 26. Accuracy medio por clases para el conjunto de datasets “imb1”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
ecoli-0vs1	96,0828	**97,5303**	96,4558	97,4512	97,3972
ecoli1	86,3680	**89,3390**	87,9042	88,5507	88,2930
ecoli2	87,4931	89,9314	90,0401	**90,1094**	90,0724
ecoli3	84,8654	**88,0955**	88,0108	86,8382	88,0090
glass-0123vs456	**93,4411**	92,7074	92,6583	93,3134	92,4624
glass0	**79,9844**	79,3014	79,5813	79,8604	79,6199
glass1	**77,9179**	73,9832	73,0879	76,8268	72,7295
glass6	**88,6153**	87,8219	86,8211	88,0247	88,3664
haberman	**56,4099**	54,9394	54,3082	55,5403	55,8133
iris0	**100,0000**	**100,0000**	**100,0000**	**100,0000**	**100,0000**
new-thyroid1	97,7222	**98,5323**	97,7815	97,3729	97,8400
new-thyroid2	97,7865	**98,0709**	96,0061	97,5499	96,2902
page-blocks0	90,9836	91,0623	90,6288	**91,1729**	91,1474
pima	67,0677	70,5787	**71,8877**	71,2958	71,8368
segment0	98,7057	97,9916	**98,7311**	98,6039	98,6142
vehicle0	91,5630	91,4320	91,3865	**91,5639**	91,2378
vehicle1	66,6008	**71,1363**	70,1794	69,4074	71,0539
vehicle2	92,0625	91,4983	**92,1312**	91,8424	91,8621
vehicle3	71,6820	72,6299	72,2825	72,8271	**73,3316**
wisconsin	96,6616	97,2955	97,2378	**97,3813**	97,3446
yeast1	64,8195	68,3896	**70,4908**	68,9474	70,1317
yeast3	87,0675	89,4290	90,6049	89,0324	**90,8842**
Media	85,1773	85,9862	85,8280	86,0687	**86,1062**

Tabla 27. Media geométrica para el conjunto de datasets “imb1”

Para el conjunto de datasets “imb1” el método que mejor resultados obtiene en media es el local, parece que en esta ocasión el método implementado es mejor que el resto. En el accuracy medio por clases obtiene casi 1 décima con respecto al método sólo local y casi 3 décimas frente al global. Para la media geométrica las diferencias son más pequeñas.

5.1.4 Conjunto “imb2”

En el conjunto “imb2” hemos aplicado lo mismo que para el conjunto “imb1”, quitamos del conjunto de train ejemplos de la clase que más tiene hasta tener el mismo número de ejemplos que la otra clase. También ejecutaremos el algoritmo 5 veces para evitar el problema de que cada ejecución del algoritmo es distinta por el hecho de quitar al azar los ejemplos.

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
abalone-17vs7-8-9-10	**78,1049**	77,0000	77,4262	77,5315	76,2333
abalone-19vs10-11-12-13	**62,7942**	56,3315	53,3172	56,3881	54,4420
abalone-20vs8-9-10	73,5460	**76,6063**	73,9894	74,7873	74,5767
abalone-21vs8	75,9651	**76,9598**	76,4333	75,3343	76,2846
abalone-3vs11	98,8700	98,2523	98,6442	**98,8906**	98,7267
abalone19	59,1677	**61,5957**	61,4720	60,8787	61,0179
abalone-9vs18	66,0182	70,2066	68,1847	70,7387	**72,6010**
car-good	**89,2810**	87,0338	88,3475	88,5336	88,2875
car-vgood	**91,3105**	89,6945	90,4331	91,0856	90,4547
cleveland-0vs4	89,3297	90,9838	90,7218	**91,3316**	90,3052
dermatology6	99,0562	**99,4978**	99,0268	98,9965	98,9083
ecoli-0-1-3-7vs2-6	81,3239	81,4330	82,4653	83,1199	**83,6653**
ecoli-0-1-4-6vs5	88,2692	**90,2692**	87,5769	88,1538	87,9615
ecoli-0-1-4-7vs2-3-5-6	84,3835	**89,8854**	88,3065	87,4209	89,1273

ecoli-0-1-4-7vs5-6	87,7889	**90,4847**	89,1720	88,1270	88,8150
ecoli-0-1vs2-3-5	86,9455	**88,0545**	85,4636	86,8182	86,0727
ecoli-0-1vs5	88,6818	**91,8182**	88,0909	88,2273	88,5909
ecoli-0-2-3-4vs5	89,5435	**92,0165**	88,7132	90,3679	89,2583
ecoli-0-2-6-7vs3-5	**87,1927**	82,6146	83,1488	85,9622	83,7451
ecoli-0-3-4-6vs5	**91,1622**	90,3784	89,1486	89,4189	89,0946
ecoli-0-3-4-7vs5-6	87,8518	**90,1547**	89,5735	89,0422	89,1282
ecoli-0-3-4vs5	89,2222	90,4444	90,5000	89,6667	**91,1667**
ecoli-0-4-6vs5	89,7162	**91,9174**	87,4745	90,0976	88,3934
ecoli-0-6-7vs3-5	**86,2500**	85,7000	84,1500	83,9500	83,4000
ecoli-0-6-7vs5	84,7000	**86,8000**	84,2000	84,0500	84,2000
ecoli4	93,0625	**93,3095**	92,0476	92,7068	92,3586
flare-F	78,7877	77,1479	79,9033	80,7562	**81,4226**
glass-0-1-4-6vs2	**67,9474**	66,0801	63,2828	67,0178	65,3867
glass-0-1-5vs2	**62,5161**	62,0860	61,9839	61,0914	62,1613
glass-0-1-6vs2	63,2333	60,9333	60,9524	**64,0048**	59,9476
glass-0-1-6vs5	**90,9714**	86,9143	90,2286	90,5143	89,9429
glass2	**66,9654**	62,8615	64,6936	64,9205	64,7936
glass4	82,9963	82,6488	85,0500	84,0988	**85,5463**
glass5	**87,9756**	86,7073	87,7317	87,7317	87,6341
kddcup-bufferOverflow-vs-back	**50,0000**	**50,0000**	**50,0000**	**50,0000**	**50,0000**
kddcup-guessPasswd-vs-satan	**50,0000**	**50,0000**	**50,0000**	**50,0000**	**50,0000**
kddcup-land-vs-portsweep	**50,0000**	**50,0000**	**50,0000**	**50,0000**	**50,0000**
kddcup-land-vs-satan	**50,0000**	**50,0000**	**50,0000**	**50,0000**	**50,0000**
kddcup-rootkitImap-vs-back	**50,0000**	**50,0000**	**50,0000**	**50,0000**	**50,0000**
kr-vs-kOne-vs-fifteen	**99,6262**	99,4646	99,6169	**99,6262**	99,6169
kr-vs-kThree-vs-eleven	**97,9888**	96,9376	97,6035	97,9432	97,6385
kr-vs-kZeroOne-vs-draw	**97,6387**	97,1192	96,9473	97,5994	97,0558
kr-vs-kZero-vs-eight	**95,0037**	92,3934	94,4168	94,7387	93,9165
kr-vs-kZero-vs-fifteen	**99,3767**	98,4855	98,8455	99,3074	98,9748
led7digit-0-2-4-5-6-7-8-9vs1	82,3675	85,2485	85,2083	84,7462	**85,4226**
page-blocks-1-3vs4	**93,9010**	91,1992	93,3596	93,2700	93,3826
poker-8-9vs5	**60,5805**	58,5561	56,6878	57,3024	55,7415
poker-8-9vs6	**78,9329**	74,9699	75,6904	75,8589	75,0301
poker-8vs6	**70,0525**	67,1005	64,9817	65,6804	64,4338
poker-9vs7	**88,3856**	79,3741	86,3936	86,2704	85,6729
shuttle-2vs5	**99,9204**	99,6450	**99,9204**	99,9173	99,9173
shuttle-6vs2-3	89,1818	**89,9091**	86,8636	87,7273	87,0909
shuttle-c0-vs-c4	**99,6000**	**99,6000**	**99,6000**	**99,6000**	**99,6000**
vowel0	**97,4939**	95,7796	92,4127	97,2050	95,8702
winequality-red-3vs5	70,0209	**71,6800**	65,7270	68,4468	68,3872
winequality-red-4	60,0339	63,2311	**65,0619**	63,2359	63,8740
winequality-red-8vs6	63,6464	69,0810	71,0217	71,1245	**73,8279**
winequality-red-8vs6-7	61,7123	60,5549	67,9129	67,2040	**69,7218**
winequality-white-3-9vs5	58,3761	61,5289	58,6235	59,7395	**62,7804**
winequality-white-3vs7	**68,8068**	64,7045	66,4545	66,7727	66,5909
winequality-white-9vs4	72,6534	**76,3977**	72,7102	73,3011	73,7443
yeast-0-2-5-6vs3-7-8-9	74,5468	75,3970	**77,9050**	76,2202	77,1837
yeast-0-2-5-7-9vs3-6-8	87,1046	89,1933	89,0768	88,8287	**90,5826**
yeast-0-3-5-9vs7-8	67,9872	69,9501	**70,9808**	70,0990	70,8648
yeast-0-5-6-7-9vs4	77,4618	79,4887	78,6502	77,8321	**79,9330**
yeast-1-2-8-9vs7	63,9317	65,6243	64,3397	66,0073	**67,2120**
yeast-1-4-5-8vs7	**63,2949**	60,8125	59,7626	61,5283	61,9466
yeast-1vs7	68,5499	69,6655	68,8256	**70,0375**	69,2055
yeast-2vs4	89,0159	**90,0898**	88,2995	89,8908	88,9359

yeast-2vs8	77,2459	74,8289	75,2833	76,5977	**78,0865**
yeast4	79,2768	**82,5727**	81,9905	80,7278	81,8828
yeast5	95,4375	**95,7986**	95,6875	95,5625	95,6667
yeast6	83,8501	86,0959	87,1501	85,9689	**87,1572**
Media	79,2320	79,2781	78,9845	79,4203	**79,4603**

Tabla 28. Accuracy medio por clases para el conjunto de datasets “imb2”

Dataset	**1-NN**	**3-NN**	**k-NN Global**	**k-NN sólo Local**	**k-NN Local**
abalone-17vs7-8-9-10	**77,9236**	76,6949	76,9081	77,2392	75,8612
abalone-19vs10-11-12-13	**61,6248**	55,6234	50,8886	54,7063	53,0055
abalone-20vs8-9-10	72,7567	**76,1444**	72,6619	73,8631	73,8429
abalone-21vs8	75,0526	**75,8305**	75,2899	74,0114	75,1549
abalone-3vs11	98,8585	98,2267	98,6265	**98,8792**	98,7103
abalone19	57,5205	**59,9086**	59,6108	58,9953	59,1027
Abalone-9vs18	64,9209	69,8267	66,6172	69,8547	**71,6075**
car-good	**88,6852**	86,0490	87,5443	87,8394	87,4855
car-vgood	**90,9391**	89,0858	89,9581	90,6639	89,9575
cleveland-0vs4	88,8058	90,6568	90,3062	**90,9827**	89,8643
dermatology6	99,0473	**99,4926**	99,0176	98,9872	98,8972
ecoly-0-1-3-7vs2-6	75,0122	70,4897	75,7873	76,5137	**77,1261**
ecoly-0-1-4-6vs5	87,3673	**89,4487**	86,7121	87,1907	87,0394
ecoly-0-1-4-7vs2-3-5-6	83,9929	**89,6229**	88,0321	87,1310	88,8286
ecoly-0-1-4-7vs5-6	87,5424	**90,3147**	88,9887	87,8943	88,6365
ecoly-0-1vs2-3-5	85,9183	**87,0313**	84,5160	85,8585	85,1864
ecoly-0-1vs5	88,0763	**91,4825**	87,5702	87,6434	88,1297
ecoly-0-2-3-4vs5	88,9554	**91,4919**	88,0386	89,7642	88,5968
ecoly-0-2-6-7vs3-5	**86,4116**	81,5259	82,2855	85,3052	83,0851
ecoly-0-3-4-6vs5	**90,7293**	89,9074	88,7709	88,9841	88,7757
ecoly-0-3-4-7vs5-6	87,3925	**89,6628**	89,1327	88,5596	88,7037
ecoly-0-3-4vs5	88,6919	89,8222	90,2273	89,2770	**90,9430**
ecoly-0-4-6vs5	89,1443	**91,6841**	86,8126	89,5933	87,8687
ecoly-0-6-7vs3-5	**84,9917**	82,9417	82,2807	81,1961	80,5499
ecoly-0-6-7vs5	84,3836	**86,5145**	83,8569	83,7265	83,8997
ecoly4	92,8654	**93,1249**	91,8162	92,5004	92,1288
flare-F	78,4920	76,2933	79,3104	80,4315	**80,9894**
glass-0-1-4-6vs2	**66,7406**	63,7865	59,6887	63,6785	61,6406
glass-0-1-5vs2	**60,1732**	57,8351	58,5862	58,0350	58,7102
glass-0-1-6vs2	60,2710	57,6880	58,5004	**61,4180**	57,2038
glass-0-1-6vs5	**90,4966**	86,1038	89,6830	90,0012	89,3713
glass2	**65,5755**	61,0321	62,2257	62,7428	62,3670
glass4	82,2166	81,6740	84,4176	83,4144	**84,9109**
glass5	85,5447	**85,6701**	85,2740	85,2792	85,1697
kddcup-bufferOverflow-vs-back	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kddcup-guessPasswd-vs-satan	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kddcup-land-vs-portsweep	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kddcup-land-vs-satan	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kddcup-rootkitImap-vs-back	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kr-vs-kOne-vs-fifteen	**99,6250**	99,4624	99,6157	**99,6250**	99,6157
kr-vs-kThree-vs-eleven	**97,9664**	96,8864	97,5711	97,9201	97,6069
kr-vs-kZeroOne-vs-draw	**97,6138**	97,0883	96,8963	97,5730	97,0130
kr-vs-kZero-vs-eight	**94,8657**	92,0618	94,2391	94,5797	93,7291
kr-vs-kZero-vs-fifteen	**99,3740**	98,4691	98,8257	99,3041	98,9634
led7digit-0-2-4-5-6-7-8-9vs1	80,4883	84,5203	84,4105	83,9350	**84,6066**
page-blocks-1-3vs4	**93,7103**	90,7653	93,1166	93,0389	93,1528

poker-8-9vs5	**59,1106**	56,7285	54,7420	55,0610	53,5167
poker-8-9vs6	**77,1376**	71,8309	73,0725	73,4901	72,4282
poker-8vs6	**67,2442**	60,5668	61,2220	61,9052	60,2464
poker-9vs7	**87,4785**	70,5467	85,0708	84,9442	84,1959
shuttle-2vs5	**99,9201**	99,6432	**99,9201**	99,9170	99,9170
shuttle-6vs2-3	88,0312	**88,8664**	85,4987	86,5340	85,8210
shuttle-c0-vs-c4	**99,5959**	**99,5959**	**99,5959**	**99,5959**	**99,5959**
vowel0	**97,4580**	95,6937	92,1624	97,1684	95,7900
winequality-red-3vs5	67,1176	**68,6250**	60,6035	63,3613	63,3905
winequality-red-4	59,2815	62,3758	**64,5675**	62,6358	63,3483
winequality-red-8vs6	61,9618	67,3781	69,9139	69,9013	**72,9712**
winequality-red-8vs6-7	59,4537	58,9057	66,0554	65,8209	**68,1132**
winequality-white-3-9vs5	52,1079	51,9891	49,9589	52,9919	**54,5040**
winequality-white-3vs7	**66,9834**	59,8120	59,5459	64,1795	62,4832
winequality-white-9vs4	63,1193	**67,6257**	62,6203	63,6251	64,0427
yeast-0-2-5-6vs3-7-8-9	74,4063	75,1021	**77,2016**	76,0208	76,7502
yeast-0-2-5-7-9vs3-6-8	87,0068	89,0928	88,7877	88,7208	**90,4578**
yeast-0-3-5-9vs7-8	67,5357	69,4559	**70,4603**	69,4707	70,3165
yeast-0-5-6-7-9vs4	77,2050	79,2405	78,4685	77,5957	**79,5366**
yeast-1-2-8-9vs7	62,8755	64,7656	63,0155	64,7917	**66,2035**
yeast-1-4-5-8vs7	**61,7676**	59,4091	55,9952	59,4393	58,8713
yeast-1vs7	67,7724	68,6023	66,4088	**69,0371**	67,7053
yeast-2vs4	88,8345	**89,9922**	88,0411	89,7621	88,7609
yeast-2vs8	76,1762	73,6001	72,2897	75,5324	**76,6782**
yeast4	78,8840	**82,1536**	81,6932	80,3896	81,5247
yeast5	95,3719	**95,7048**	95,6010	95,4912	95,5798
yeast6	83,4558	85,7748	86,8611	85,6662	**86,8694**
Media	74,7679	74,4523	74,1643	**74,8108**	74,7624

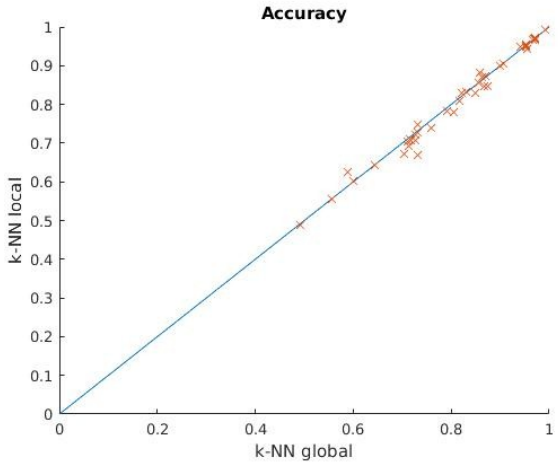
Tabla 29. Media geométrica para el conjunto de datasets “imb2”

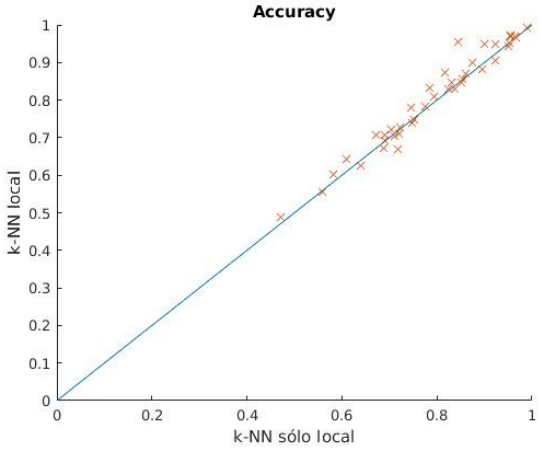
Para el accuracy medio por clases, el método con mejor media es el k-NN local y para la media geométrica el método k-NN sólo local. Parece que ambos métodos mejorarán al global, sin embargo es posible que entre ellos no encontremos diferencias estadísticas. Más adelante con los test lo comprobaremos.

5.2 Gráficas

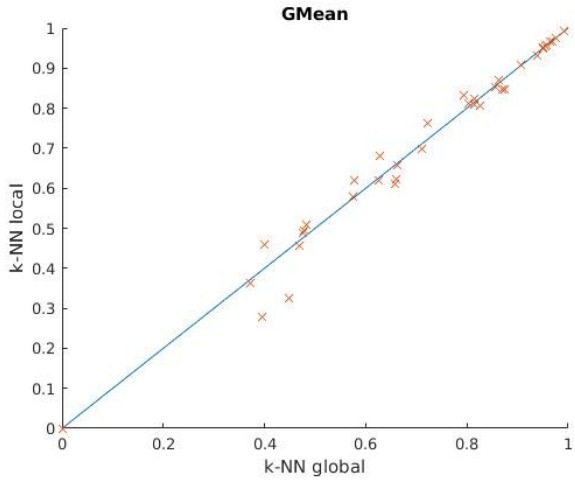
Mostraremos gráficas de unos métodos frente a otros para tener una idea más aproximada de cuál de los métodos es mejor o lo parece, ya que hasta que no hagamos los test no podremos afirmar nada ni en un sentido ni en otro. Para los datasets normales mostraremos la gráfica para el accuracy y la media geométrica comparando el método local implementado frente al global y al sólo local.

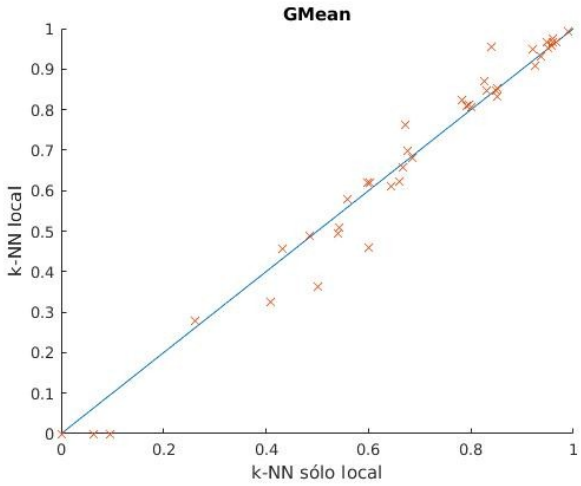
5.2.1 Gráficas del conjunto “small”





Figuras 4 y 5. Gráficas comparando el método local frente al global y sólo local para el accuracy con el conjunto de datasets “small”

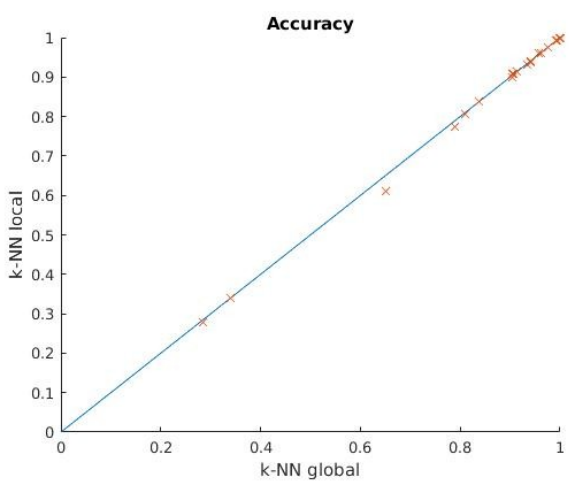


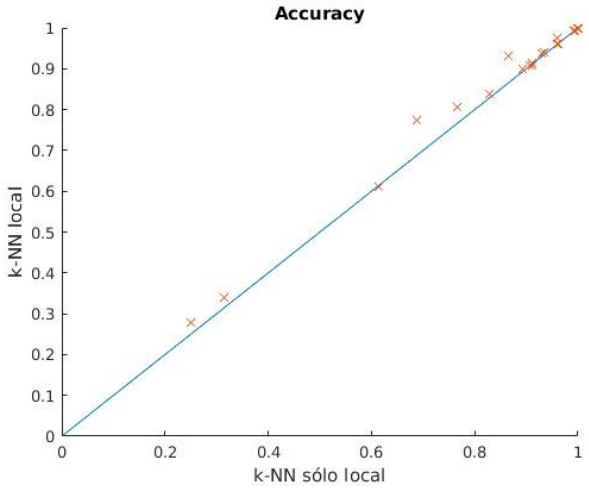


Figuras 6 y 7. Gráficas comparando el método local frente al global y sólo local para la media geométrica con el conjunto de datasets “small”

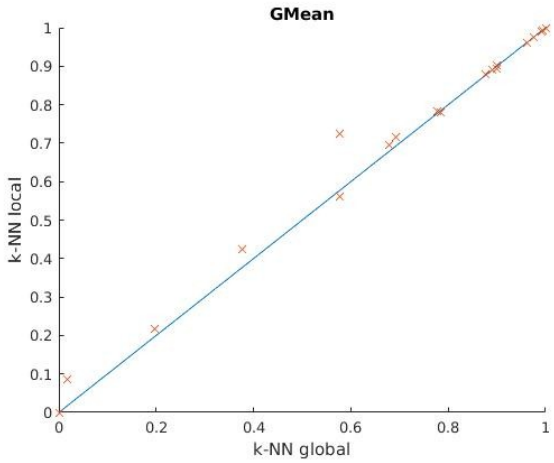
En el caso del método local frente al global parece que la diferencia entre uno y otro es pequeña, para el accuracy sí que claramente obtiene mejores resultados el global pero con la media geométrica no está tan claro. En la comparación del método local frente al sólo local parece que para el accuracy el método local será mejor aunque para la media geométrica es algo más difícil pronosticar cuál ganará.

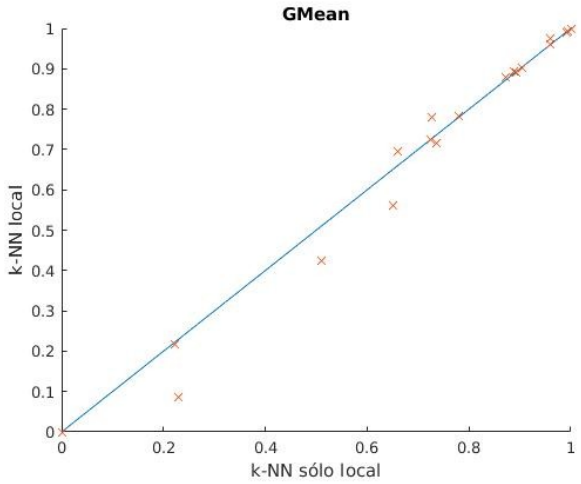
5.2.2 Gráficas del conjunto “big”





Figuras 8 y 9. Gráficas comparando el método local frente al global y sólo local para el accuracy con el conjunto de datasets “big”

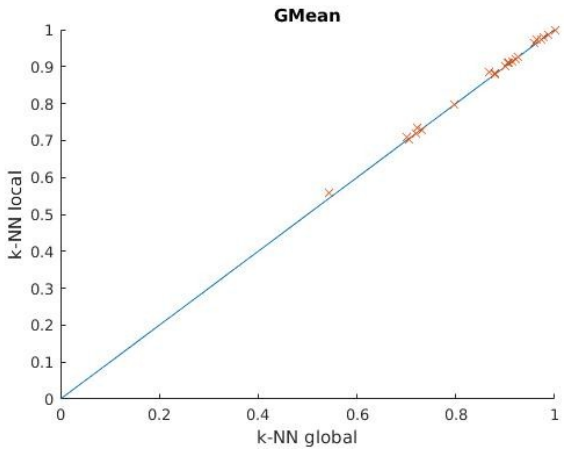


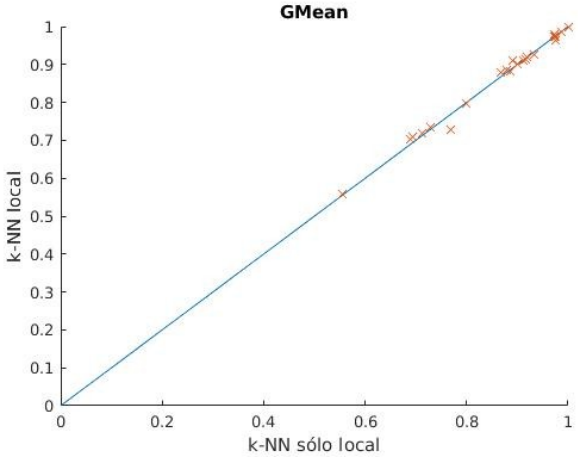


Figuras 10 y 11. Gráficas comparando el método local frente al global y sólo local para la media geométrica con el conjunto de datasets “big”

Por las gráficas podemos ver que el test de Wilcoxon será muy igualado en la comparación entre el método global y el local, apenas hay diferencias claras entre victoras-derrotas. En la comparación local contra sólo local parece que el local será mejor al menos para el accuracy, la media geométrica está más igualada.

5.2.3 Gráficas del conjunto “imb1”

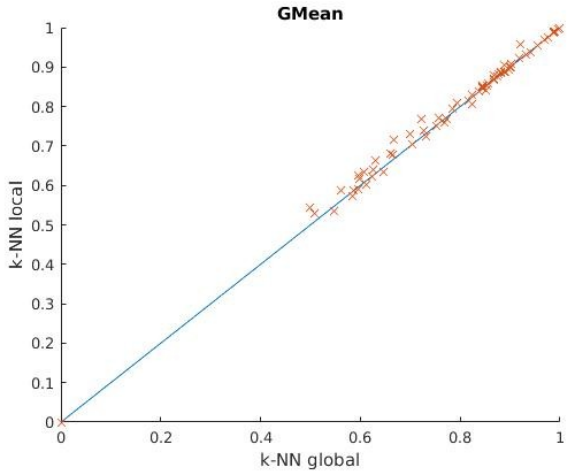


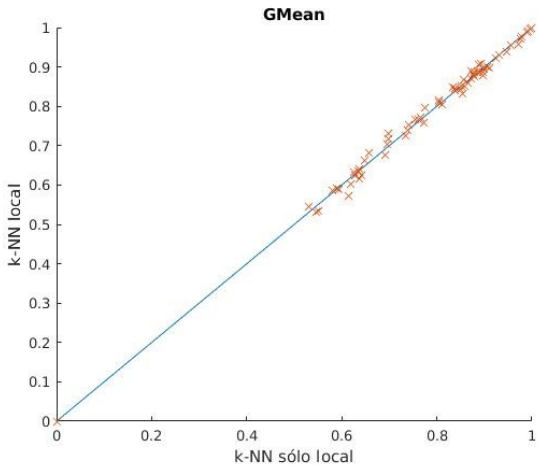


Figuras 12 y 13. Gráficas comparando el método local frente al global y sólo local para la media geométrica con el conjunto de datasets “imb1”

Los resultados con los datasets imbalanceados del conjunto imb1 son muy parecidos en las comparaciones. Las diferencias son muy pequeñas entre los métodos, en la comparación local contra global el local parece ser ligeramente mejor y en la comparación local contra sólo local es éste último el que parece mejor.

5.2.4 Gráficas del conjunto “imb2”





Figuras 14 y 15. Gráficas comparando el método local frente al global y sólo local para la media geométrica con el conjunto de datasets “imb2”

En la gráfica de la comparación local contra global, el local gana unas cuantas veces más, en la gráfica comparando el método local frente al sólo local están muy igualadas las victorias-derrotas.

5.3 Test estadísticos

Los test nos permitirán asegurar que un método es mejor que otro u otros. Realizaremos primero el test de rangos alineados de Friedman para comparar los 5 métodos a la vez. Después utilizaremos el test post-hoc de Hommel para comprobar si hay diferencias significativas entre el método con mejor rango y el resto. En caso de que un método que no sea el método local obtenga el mejor rango compararemos ese método con el local mediante el test de Wilcoxon. Si con el test de Friedman el método con mejor rango es el local pero no hay diferencias significativas con los demás, compararemos al método local frente al método que haya obtenido en segundo lugar el mejor rango.

También mostraremos al final una tabla de tiempos con el tiempo mínimo, medio y máximo para los 3 métodos que mejor resultados han dado en los test para los 4 conjuntos de datasets. Esta tabla de tiempos nos permitirá decidir qué método es mejor en los casos en que no hay diferencias significativas entre ellos.

5.3.1 Test para el conjunto “small”

Mostraremos el test de rangos alineados de Friedman y el de Wilcoxon si fuera necesario, realizado sobre los resultados de accuracy y media geométrica para el conjunto de datasets “small”.

Rangos alineados de Friedman (Acc)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
157,54	130,46	**60,06**	100,02	66,91

Tabla 30. Test de rangos alineados de Friedman para el accuracy con el conjunto de datasets “small”

Test de Hommel	
Algoritmo	APV
1-NN	4,04E-13
3-NN	2,32E-07
k-NN sólo Local	4,57E-03
k-NN Local	6,01E-01

Tabla 31. Test post-hoc de Hommel

Podemos afirmar que el método global es mejor estadísticamente que todos los otros métodos excepto el local. Compararemos mediante el test de Wilcoxon el método local y el global.

Test de Wilcoxon (Acc)		
Algoritmo	k-NN Global	k-NN Local
rango	**597,5**	263,5
p	0,0282	

Tabla 32. Test de Wilcoxon comparando el método local y el global para el accuracy

El método global obtiene un rango mayor y el p valor calculado es menor que 0’05, por tanto podemos afirmar que es mejor estadísticamente que el local.

Los test para la media geométrica:

Rangos alineados de Friedman (GM)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
114,90	111,54	**93,82**	97,06	97,68

Tabla 33. Test de rangos alineados de Friedman para la media geométrica con el conjunto de datasets “small”

Test de Hommel	
Algoritmo	APV
1-NN	3,52E-01
3-NN	5,29E-01
k-NN sólo Local	8,04E-01
k-NN Local	8,04E-01

Tabla 34. Test post-hoc de Hommel

El método global obtiene un mejor rango pero el test de Hommel nos muestra que no hay diferencias estadísticas entre los métodos. Compararemos con el test de Wilcoxon el método local frente al global.

Test de Wilcoxon (GM)		
Algoritmo	k-NN Global	k-NN Local
rango	**487**	374
p	0,6106	

Tabla 35. Test de Wilcoxon entre el método local y el global para la media geométrica

El método global obtiene un mejor rango, pero el p valor no es lo suficientemente pequeño como para poder afirmar que sea mejor. Como ya veíamos en las gráficas, para el accuracy el global se intuía claramente mejor y con la media geométrica parecía que iba a estar algo más reñido.

5.3.2 Test para el conjunto “big”

Mostraremos igual que antes el test de rangos alineados Friedman y el de Wilcoxon sobre los resultados de accuracy y media geométrica.

Rangos alineados de Friedman (Acc)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
77,05	61,65	**28,55**	55,83	29,42

Tabla 36. Test de rangos alineados de Friedman para el accuracy con el conjunto de datasets “big”

Test de Hommel	
Algoritmo	APV
1-NN	4,99E-07
3-NN	9,26E-04
k-NN sólo Local	5,90E-03
k-NN Local	9,24E-01

Tabla 37. Test post-hoc de Hommel

Mismos resultados que con el conjunto de datasets “small”, el método global obtiene el mejor rango y el test de Hommel nos permiten afirmar que es mejor método que todos los

demás excepto el local, haremos por tanto el test de Wilcoxon entre el método global y el local.

Test de Wilcoxon (Acc)		
Algoritmo	k-NN Global	k-NN Local
rango	**158,5**	51,5
p	0,0486	

Tabla 38. Test de Wilcoxon entre el método local y el global para el accuracy

El método global obtiene mejor rango y el p valor calculado es menor o igual a 0’05, por tanto es mejor estadísticamente que el local.

Los test para la media geométrica:

Rangos alineados de Friedman (GM)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
55,85	54,70	55,50	**39,62**	46,82

Tabla 39. Test de rangos alineados de Friedman para la media geométrica con el conjunto de datasets “big”

Test de Hommel	
Algoritmo	APV
1-NN	1,54E-01
3-NN	2,01E-01
k-NN Global	1,67E-01
k-NN Local	4,33E-01

Tabla 40. Test post-hoc de Hommel

A pesar de que el método con mejor rango sea el sólo local, no podemos afirmar que éste sea mejor que ningún otro puesto que el test de Hommel no encuentra diferencias significativas entre ellos. Compararemos el método local frente al sólo local mediante Wilcoxon.

Test de Wilcoxon (GM)		
Algoritmo	K-NN sólo Local	k-NN Local
rango	**118**	92
p	0,6417	

Tabla 41. Test de Wilcoxon comparando el método sólo local frente a local para la media geométrica

Tampoco podemos afirmar que un método sea mejor que el otro a pesar de que es el sólo local quien gana más veces y por tanto obtiene un rango más alto.

5.3.3 Test para el conjunto “imb1”

Como ya hemos dicho para los datasets imbalanceados lo que más nos interesa medir es la media geométrica, por tanto mostraremos los test de rangos alineados de Friedman y Wilcoxon si fuese necesario, sobre los resultados de la media geométrica para el conjunto de datasets “imb1”.

Rangos alineados de Friedman (GM)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
67,86	55,41	60,55	48,14	**45,55**

Tabla 42. Test de rangos alineados de Friedman para la media geométrica con el conjunto de datasets “imb1”

Test de Hommel	
Algoritmo	APV
1-NN	8,12E-02
3-NN	6,10E-01
k-NN Global	3,57E-01
k-NN sólo Local	7,88E-01

Tabla 43. Test post-hoc de Hommel

Con el test de rangos alineados de Friedman no podemos afirmar que haya un método mejor que los otros. Compararemos mediante Wilcoxon el método local con el sólo local; por ser el segundo con mejor rango, y con el global; por comprobar la diferencia en datasets imbalanceados entre un método que utiliza la misma k para todo el conjunto y otro que puede utilizar distintos valores de k al clasificar.

Test de Wilcoxon (GM)		
Algoritmo	k-NN sólo Local	k-NN Local
rango	107,5	**145,5**
p	0,5202	

Test de Wilcoxon (GM)		
Algoritmo	k-NN Global	k-NN Local
rango	74,5	**178,5**
p	0,0853	

Tablas 44 y 45. Test de Wilcoxon comparando el método local frente al global y el sólo local para la media geométrica

Una vez más el método local es el que obtiene el mayor rango. El p valor calculado no permite afirmar que sea mejor que los otros 2 al 95% de confianza, sin embargo al 90% de confianza sí que sería mejor que el global, por lo tanto se podría decir que sí es mejor que el global. Viendo las gráficas del conjunto “imb1” ya dijimos que las diferencias eran muy pequeñas y así se ha demostrado.

5.3.4 Test para el conjunto “imb2”

Mostraremos al igual que para el conjunto “imb1” el test de rangos alineados de Friedman y Wilcoxon para la media geométrica.

Rangos alineados de Friedman (GM)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
178,15	178,10	211,10	**165,56**	182,10

Tabla 46. Test de rangos alineados de Friedman para la media geométrica con el conjunto de datasets “imb2”

Test de Hommel	
Algoritmo	APV
1-NN	4,73E-01
3-NN	4,73E-01
k-NN Global	3,65E-02
k-NN Local	4,73E-01

Tabla 47. Test post-hoc de Hommel

En esta ocasión el método sólo local es el que obtiene un mejor rango. A partir de los p valores del test de Hommel sólo podemos asegurar que es mejor que el método global. Compararemos como hemos hecho con el conjunto “imb1” el método local frente al sólo local y al global.

Test de Wilcoxon (GM)		
Algoritmo	k-NN sólo Local	k-NN Local
rango	**1421**	1280
p	0,7518	

Test de Wilcoxon (GM)		
Algoritmo	k-NN Global	k-NN Local
rango	816	**1885**
p	2,60E-03	

Tablas 48 y 49. Test de Wilcoxon comparando el método local frente al sólo local y al global para la media geométrica

Como ya vimos en las gráficas de este conjunto de datasets, el método local obtenía muchas más victorias que el global y de hecho es mejor estadísticamente según demuestra el test. En la comparación entre el local y el sólo local la gráfica se veía muy igualada, obtiene mejor rango el sólo local pero el p valor no es lo suficientemente bajo.

5.3.5 Tablas de tiempos

Los métodos que mejores resultados han dado han sido el global, el sólo local y el local, por tanto mostraremos la tabla de tiempos de estos 3 métodos para todos los conjuntos de datasets. El tiempo medido es desde que comienza el aprendizaje de la k hasta que finaliza la clasificación del conjunto de test. Dado que el proceso de clasificación una vez que tenemos la k es igual en los 3 métodos, las diferencias que haya serán las diferencias que hay en el aprendizaje de la k, ya que el tiempo de test en los 3 será prácticamente el mismo.

	Conjunto “small”			**Conjunto “big”**		
	Mínimo	**Media**	**Máximo**	**Mínimo**	**Media**	**Máximo**
k-NN Global	0,07	0,36	3,97	2,03	41,44	413,85
k-NN sólo Local	0,11	0,77	6,35	5,36	53,73	483,56
k-NN Local	0,18	1,07	9,53	6,83	84,79	793,72

Tabla 50. Tabla de tiempos de los métodos global, sólo local y local para los conjuntos de datasets normales

	Conjunto “imb1”			**Conjunto “imb2”**		
	Mínimo	**Media**	**Máximo**	**Mínimo**	**Media**	**Máximo**
k-NN Global	0,05	0,20	0,66	0,01	0,06	0,17
k-NN sólo Local	0,09	0,39	1,73	0,02	0,08	0,26
k-NN Local	0,15	0,57	2,21	0,02	0,13	0,41

Tabla 51. Tabla de tiempos de los métodos global, sólo local y local para los conjuntos de datasets imbalanceados

Comprobamos que el método global es el más rápido de los 3 y que el local es prácticamente una suma del global y el sólo local, algo normal dado que lo que hace precisamente éste método es calcular un accuracy global y otro local y sacar el valor de k local en función de éstos, mientras que los otros 2 métodos sólo realizan una de esas 2 partes.

Para los conjuntos de datasets imbalanceados comprobamos que los tiempos en general son inferiores a los obtenidos con los conjuntos normales, algo totalmente razonable ya que como hemos dicho hacemos undersampling para igualar el número de ejemplos de ambas clases en el conjunto de train antes de comenzar, lo que hace que nuestros conjuntos de train se queden muy reducidos.

5.4 Discusión

Como hemos podido comprobar con los test, el método k-NN local explicado anteriormente no es todo lo bueno que debería.

Para el conjunto de datasets normales (small y big) no es el método que mejor rango obtiene en el test de rangos alienados de Friedman, sino que es el global quién obtiene el mejor rango y mejora estadísticamente a todos los demás para el accuracy. Además es el más rápido, por lo que es el mejor método a utilizar con este tipo de datasets.

Para los conjunto de datasets imbalanceados obtenemos mejores resultados en cambio. Con el conjunto “imb1” el método con mejor rango en el test de rangos alineados de Friedman es el local, el test de Wilcoxon muestra que no hay diferencias significativas con el global y el sólo local, aunque como hemos dicho, el p valor obtenido en la comparación con el global es lo suficientemente pequeño como para poder decir que es mejor. Con el conjunto “imb2” el método local mejora al global pero el método con mejor rango para el test de rangos alineados de Friedman es el sólo local. Al ser el local y el sólo local los mejores métodos para los datasets imbalanceados y no haber diferencias significativos entre ellos la única forma que tenemos de decantarnos entre uno y otro es mediante la tabla de tiempos. El sólo local es más rápido de modo que sería la mejor opción.

Este método no parece ser muy bueno para datasets normales, sin embargo para los datasets imbalanceados sí que parece un buen método a utilizar. Tanto el sólo local como el local demuestran tener mejores resultados que el global con este tipo de datasets. A priori parecía normal que en datasets imbalanceados fuese mejor un método que calcula la k específicamente para cada prototipo, que otro que calcula la k, que es única, en función del rendimiento del valor de k en todo el conjunto de train. Con los test se ha comprobado que es así, el método global es el peor de los 3 métodos en los 2 conjuntos imbalanceados.

Para comprobar que el método se ha implementado correctamente probamos a ejecutar el programa pero esta vez clasificando el propio conjunto de train del que aprendemos la k en vez del conjunto de test. Lo hicimos para los 4 conjuntos de datasets pero sólo mostraremos para el conjunto “small”, ya que todos los resultados son similares.

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	80,4706	84,4932	86,1957	**90,7971**	90,4709
balance	77,3621	82,6796	89,4003	91,6401	**91,7599**
bands	70,8219	69,9315	71,2329	85,6164	**85,6849**
breast	67,1368	69,6699	74,7305	**82,0360**	81,5848
bupa	60,5072	61,5942	63,4783	**84,1304**	**84,1304**
car	87,7749	92,8820	94,2855	98,3363	**98,4375**
chess	89,6198	94,6105	94,8609	**98,1931**	98,1539
cleveland	53,0319	54,7967	58,3332	69,4417	**69,6949**
contraceptive	42,9571	45,4349	48,6929	69,6709	**69,6880**
crx	81,3942	84,4560	87,2892	**89,8924**	89,5480
dermatology	95,3894	96,5768	96,6467	99,1603	**99,1613**
ecoly	77,8980	80,0726	80,5160	**88,9167**	88,6198
flare	63,2320	61,8791	57,4148	76,9077	**77,0250**
german	68,7500	69,9000	72,8500	**83,4500**	83,2500
glass	69,5000	67,6339	69,8529	**81,3086**	80,6088
haberman	65,6869	69,0365	75,1639	80,8816	**81,2907**
hayes-roth	33,4372	27,1906	33,4372	**76,7121**	75,7721
heart	75,7407	78,2407	82,9630	88,4259	**88,6111**
housevotes	91,2750	91,5988	91,9225	**94,2893**	94,0748
ionosphere	86,1108	85,1136	85,9685	**90,7407**	**90,7407**
iris	95,3333	95,0000	96,6667	**98,1667**	**98,1667**
led7digit	28,8691	31,4990	41,8148	**51,9295**	51,6285
mammographic	75,0309	79,8802	81,6880	**87,2301**	87,2000
monk-2	76,7931	94,4435	94,4435	**97,6270**	**97,6270**
movement-libras	80,9563	75,5142	80,9563	**87,1954**	86,6443
newthyroid	96,0465	93,9535	96,2791	**97,0930**	**97,0930**
pima	70,1162	73,3069	74,9348	**86,0676**	86,0674
saheart	64,7165	67,4240	72,3495	**81,4385**	81,1143
segment	96,4827	95,2273	96,4827	**98,1277**	98,1169
sonar	85,2108	81,7305	84,9713	94,7132	**94,8322**
spectfheart	69,7560	71,9113	80,0575	88,5771	**88,8596**
splice	74,5299	76,1600	87,1786	94,9530	**94,9687**
tic-tac-toe	80,6111	85,1510	86,2211	**92,2755**	91,8320
titanic	70,8690	70,8974	72,4010	**75,0245**	**75,0245**
vehicle	69,2084	70,4788	70,8640	85,4616	**85,6092**
vowel	98,8889	95,8333	98,8889	**99,2929**	99,0909
wdbc	95,7816	96,7045	96,9242	**98,3743**	98,3741
wine	95,6395	96,3507	97,4706	**99,2997**	99,1588
winequality-red	52,0792	51,6260	56,4106	75,2820	**75,4843**
wisconsin	95,9006	96,9987	97,3649	**98,0602**	97,9871
yeast	50,3029	52,3757	58,3236	**73,6022**	73,5009
Media	74,6639	76,1038	78,9738	**87,3253**	87,2363

Tabla 52. Accuracy sobre train para el conjunto de datasets “small”

Dataset	1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
australian	80,1753	84,3151	86,2330	**90,6580**	90,4161
balance	0,0000	0,0000	0,0000	25,9750	**26,0135**
bands	67,4456	66,0409	67,1761	**83,7157**	82,8629
breast	55,0932	54,5175	50,9947	**66,4961**	65,1574
bupa	58,7861	59,8804	57,4611	**83,6459**	82,7197
car	68,2446	78,4216	73,0344	**91,8410**	91,5768
chess	89,2760	94,4336	94,6798	**98,1277**	98,0799
cleveland	6,4161	0,0000	0,0000	**8,7493**	8,4099
contraceptive	41,4804	41,3429	46,7110	66,7245	**66,9609**
crx	80,8164	84,2678	87,5285	**89,9524**	89,6588
dermatology	95,0414	96,3870	96,1718	99,1160	**99,1289**
ecoly	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
flare	40,4189	34,9342	24,6272	**54,7500**	53,6454
german	59,4971	55,7025	46,6299	**71,4154**	68,9930
glass	60,3233	31,2041	37,0585	**65,7735**	54,6672
haberman	49,5213	43,5314	41,8220	**67,1302**	63,0397
hayes-roth	34,4416	26,3499	34,4416	**74,7732**	73,1662
heart	75,3961	77,7907	81,8791	88,0241	**88,1294**
housevotes	91,4511	91,7930	92,1103	**94,4444**	94,2349
ionosphere	80,1775	78,0859	79,9334	**87,0402**	**87,0402**
iris	95,2606	94,9186	96,6144	**98,1453**	**98,1453**
led7digit	0,0000	0,0000	33,3495	**40,4028**	33,5077
mammographic	74,9660	79,7771	81,6861	**87,1766**	87,1774
monk-2	75,1665	94,5415	94,5415	**97,7108**	97,7069
movement-libras	79,0985	72,5843	79,0985	**86,1937**	85,5450
newthyroid	94,1014	88,7157	94,6249	**95,5477**	**95,5477**
pima	65,1777	67,8336	64,7214	**81,4548**	81,3477
saheart	57,6270	58,8695	57,8391	**73,5700**	72,3453
segment	96,3931	95,0794	96,3931	**98,1037**	98,0926
sonar	84,7808	81,0047	84,8655	94,5619	**94,6389**
spectfheart	54,1551	50,5493	38,7922	**78,3905**	74,2691
splice	75,0936	77,3420	86,6112	**95,2087**	95,1591
tic-tac-toe	70,3433	78,5179	80,0187	**88,6906**	88,0232
titanic	48,9320	52,4914	56,2906	**57,7309**	**57,7309**
vehicle	66,0806	66,9548	66,6723	84,3184	**84,4087**
vowel	98,8828	95,7488	98,8828	**99,2884**	99,0854
wdbc	95,4524	96,0737	96,2957	**97,9463**	97,8449
wine	96,2029	96,7121	97,8398	**99,4082**	99,2906
winequality-red	**0,0000**	**0,0000**	**0,0000**	**0,0000**	**0,0000**
wisconsin	95,2546	96,7450	97,1986	**98,0244**	97,9194
yeast	27,1632	0,0000	0,0000	**34,8247**	34,5121
Media	63,0277	62,0356	63,4348	**75,4891**	74,5414

Tabla 53. Media geométrica sobre train con el conjunto de datasets “small”

Como se puede comprobar la diferencia es muy grande entre los métodos local y sólo local frente a los otros 3 en media cuando clasificamos el conjunto train. En los resultados que hemos enseñado más atrás, para el accuracy las medias de los 5 métodos estaban entre 75 y 81. En este caso las medias de los métodos 1-NN, 3-NN y k-NN global se mantienen parecidas a ese rango pero los del método local y sólo local se disparan siendo mucho mejores. Haremos también los test.

Rangos alineados de Friedman (Acc)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
161,22	149,67	119,21	**42,07**	42,83

Tabla 54. Test de rangos alineados de Friedman para el accuracy sobre train con el conjunto “small”

Test de Hommel	
Algoritmo	APV
1-NN	3,83E-19
3-NN	6,52E-16
k-NN Global	7,86E-09
k-NN Local	9,54E-01

Tabla 55. Test post-hoc de Hommel

Tanto el método local como el sólo local obtienen un rango mucho mejor que los otros 3. Compararemos ambos mediante Wilcoxon.

Test de Wilcoxon (Acc)		
Algoritmo	k-NN sólo Local	k-NN Local
rango	**556,5**	304,5
p	0,0855	

Tabla 56. Test de Wilcoxon comparando el método local y el sólo local para el accuracy sobre train

El test muestra que el método sólo local gana en muchos datasets al local pero no lo suficiente como para afirmar que haya diferencias estadísticas entre ellos.

Haciendo los mismos test sobre la media geométrica los resultados son parecidos.

Rangos alineados de Friedman (GM)				
1-NN	3-NN	k-NN Global	k-NN sólo Local	k-NN Local
142,04	149,17	133,50	**43,54**	46,76

Tabla 55. Test de rangos alineados de Friedman para la media geométrica sobre train con el conjunto “small”

Test de Hommel	
Algoritmo	APV
1-NN	1,67E-13
3-NN	2,99E-15
k-NN Global	1,32E-11
k-NN Local	8,06E-01

Tabla 56. Test post-hoc de Hommel

Test de Wilcoxon (GM)		
Algoritmo	k-NN sólo Local	k-NN Local
rango	**752,5**	108,5
p	2,21E-05	

Tabla 56. Test de Wilcoxon comparando el método local frente al sólo local para la media geométrica sobre train

En esta ocasión el método sólo local incluso mejora al local y ambos siguen siendo muy superiores a los otros 3.

Pese a que sabemos que estos test sobre el conjunto de train no sirven para comparar la calidad de los métodos sí que nos demuestran que el método lo hemos entendido bien y está

bien implementado y que la razón de que los resultados sobre test sean tan distintos pueden deberse al sobre-entrenamiento.

6. Conclusiones y Líneas Futuras

En este trabajo hemos implementado una variante del algoritmo k-NN estándar, que consiste en obtener a partir del conjunto de train, un valor local de k que clasifique mejor el vecindario de cada prototipo y clasificar cada nuevo ejemplo utilizando para ello tantos vecinos como indique el valor local de k de su prototipo más cercano. Sin embargo este método no ha demostrado tener un mejor rendimiento que el resto de métodos con los que lo hemos comparado (1-NN, 3-NN, k-NN global y k-NN sólo local), o al menos no en todos los casos. Para los datasets normales es peor que el método global, pero con los datasets imbalanceados demuestra ser junto al sólo local un método mejor que el global.

Para comprobar que el método estaba bien implementado ejecutamos el programa clasificando el conjunto de train en vez del de test y los resultados entonces sí que dieron mejor rendimiento, lo cual nos indica que el método sobre-entrena en exceso. El método local no pasaba a ser el mejor porque daba resultados muy parecidos al sólo local o incluso algo peor, pero la diferencia entre estos 2 métodos y los más clásicos y simples (1-NN, 3-NN y el k-NN global, que no es más que quedarse con el mejor valor de k de una validación cruzada) se hizo muy grande con una diferencia en media de casi 10 puntos. Por tanto, se ve que el método expuesto sobre-entrena en exceso al calcular la k local asociada a cada prototipo y ése valor de k calculado sólo es tan bueno cuando estamos clasificando al propio conjunto de train.

Otra posibilidad es que el método sólo sea mejor en ciertas bases de datos. Con nuestras bases de datos no hemos conseguido replicar los resultados que se muestran en el artículo a partir del que hemos implementado este método. El artículo no da información de las bases de datos que utiliza para las pruebas más allá del rango de número de ejemplos y atributos por lo que no hemos podido identificarlas.

Ésta sería la línea futura de investigación, podríamos coger las bases de datos en las cuales nuestro método obtiene un mejor rendimiento e intentar sacar similitudes entre ellas y reglas en base al número de clases, número de atributos o número de ejemplos que nos permita decir cuáles son las bases de datos para las que este método es el mejor. Este estudio tendría sentido especialmente con los datasets normales, ya que con los imbalanceados sí que éste método parece ser bueno en general para cualquier datasets.

Con la explicación que aparece en el documento del que hemos obtenido éste método no cabe otra implementación posible del algoritmo. Por tanto, pensamos que la diferencia está en las bases de datos utilizadas para medir el rendimiento del método.

7. Referencias

[1] N. García-Pedrajas, J. A. Romero del Castillo and G. Cerruela-García, "A Proposal for Local k Values for k -Nearest Neighbor Rule," in IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 2, pp. 470-475, Feb. 2017.

[2] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesus, L. Sánchez, F. Herrera. KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining International Journal of Computational Intelligence Systems 10 (2017) 1238-1249, url: http://keel.es