uc3m | Universidad Carlos III de Madrid

Máster Universitario en Ingeniería Matemática
2018-2019

Trabajo Fin de Máster

Procedimiento de vecinos más cercanos con matrices de distancias parcialmente observadas

---

Aldo Ramón Franco Comas

Tutor
Andrés M. Alonso Fernández

Leganés, 2019

ABSTRACT

The nearest neighbors procedure is a non-parametric procedure that is used for the classification of new observations using the distance matrix between the observations to be classified and the observations in the training sample. In this project, we obtain a k-NN procedure when for reasons of response time, computational cost (e.g., in extremely large datasets) or destructive test it is not possible to calculate the distances between the new observations and all the observations in the training sample. The cases in which the distance matrix in the training sample is completely or partially observed will be addressed. The proposed procedure is illustrated using two well known real datasets.

Key Words: Distance matrix, k-NN, ultrametric, additive and cluster.

AGRADECIMIENTOS

Quisiera agradecer a mis tíos por haberme apoyado en todo momento en esta aventura. A mis padres por darme ánimo a pesar de la distancia y aceptar cada decisión que he tomado. A mi primo que al final se ha convertido en un hermano menor y a las abuelas que he conocido por asumirme como si fuera otro nieto. También agradecer a AXA Seguros S.A. por permitirme usar ese magnífico clúster que tienen en su sede en Francia.

A todos, muchas gracias.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

1. INTRODUCCIÓN

1.1. Motivación del trabajo

Desde tiempos remotos los conceptos de distancia y cercanía han estado presentes en nuestra vida diaria. Este es uno de los motivos por el cual el algoritmo k-NN (k-Nearest Neighbors o k-Vecinos más Cercanos) sea fácil de entender intuitivamente, incluso por personas que tengan poca base matemática. Es uno de los métodos no paramétricos de aprendizaje supervisado más simple. En lo que sigue, se presentan las características básicas del procedimiento k-NN.

Los datos de entrada tienen la forma $\{(x_i, y_i)\}$ donde x_i es un vector de longitud p, donde están representados p atributos para cada caso i, mientras que y_i es la variable respuesta o etiqueta que toma el caso i, dicha variable puede ser un valor numérico como por ejemplo la edad, el salario, la estatura o el peso; también puede ser un valor categórico como el género, el tipo de contrato laboral o la tipología de riesgo de accidentes. Igualmente se necesita una distancia entre un caso x_i y otro x_j denotada por $d(x_i, x_j)$. Para cada nuevo caso x_0 se calculan las distancias $d(x_0, x_i)$ para todo $i = 1, ..., n$, siendo n el total de casos en el conjunto de entrenamiento y se buscan los k casos más cercanos.

- En la clasificación k-NN, la salida generalmente es una clase. Al nuevo objeto se le asigna la clase más común entre sus k vecinos más cercanos. Si $k = 1$, entonces el objeto simplemente se asigna a la clase de ese vecino más cercano.
- En la regresión k-NN, la salida usualmente es el promedio de los valores de los k vecinos más cercanos.

Existen otros procedimientos de asignación/clasificación y predicción como vecinos más cercanos ponderados donde se da mayor peso a estos últimos. Más detalles en [1].

Como ventajas principales del k-NN tenemos:

- No paramétrico: No hace suposiciones explícitas sobre la forma funcional de los datos, evitando los peligros de alejarse de la distribución subyacente de los mismos.
- Algoritmo simple: Fácil de explicar y de comprender.
- Alta precisión: Es bastante alta, sin embargo puede no ser competitiva en comparación con otros métodos de aprendizaje supervisado, como pueden ser la redes neuronales o las máquinas de vector soporte (SVM), que utilizan procesos de entrenamiento mucho más costosos computacionalmente.
- Evoluciona constantemente: Dado que es un aprendizaje basado en casos el algoritmo se adapta inmediatamente a medida que recopilamos nuevos datos de entrenamiento. Esto permite que el algoritmo responda rápidamente a los cambios en la entrada durante el uso en tiempo real.
- Fácil de implementar para problemas de múltiples clases: La mayoría de los algoritmos son fáciles de implementar para problemas binarios sin embargo requieren

un esfuerzo extra a la hora de ser implementados para múltiples clases, como puede ser el caso de las SVM, mientras que k-NN se ajusta a múltiples clases sin ningún esfuerzo adicional.

- Clasificación y regresión: Una de las mayores ventajas del k-NN es que se puede usar tanto para problemas de clasificación como de regresión.
- Variedad de distancias: Ofrece una alta flexibilidad a la hora de elegir cualquier distancia al construir el modelo.

Mientras que sus desventajas son:

- Datos no balanceados: No funciona bien en datos no balanceados. Si consideramos un problema de clasificación binario con las siguientes clases rojo y azul, y la mayoría de los datos de entrenamiento están etiquetados como rojo, entonces el modelo pronosticará casi siempre rojo. Esto podría resultar en la clasificación errónea de la clase menos representada.
- Sensibilidad a valores atípicos: Es sensible a los valores atípicos, ya que simplemente elige a los vecinos según los criterios de distancia.
- Tratamiento con valores perdidos: No tiene la capacidad de lidiar con el problema de valores perdidos en las variables independientes o regresoras.
- Computacionalmente costoso: Puede ser muy fácil de implementar, pero a medida que crece el conjunto de datos, la velocidad del algoritmo disminuye rápidamente.
- Necesita características homogéneas: Es recomendable que las características tengan la misma escala pues si una variable está dada en milésimas de unidades y el resto en unidades, esta variable tendrá un impacto muy grande a la hora de calcular las distancias.
- Maldición de la dimensión: Bajo un amplio conjunto de condiciones, a medida que se aumenta la dimensionalidad, la distancia al punto de datos más cercano se acerca a la distancia al punto de datos más lejano, ver en [2].

Dado lo anterior se puede llegar a la conclusión de que en problemas donde n es muy grande, como ocurre hoy día en la era del Big Data, calcular todas estas distancias para predecir no sea factible. Esta situación puede darse, por ejemplo, cuando, por razones de tiempo de respuesta, coste computacional (por ejemplo, en conjuntos de datos extremadamente grandes) o pruebas destructivas (análisis de ADN), no es posible calcular todas las distancias de las nuevas observaciones a todas las observaciones en la muestra de entrenamiento. En este trabajo, propondremos una solución para los casos en que no sea posible calcular todas las distancias del caso a clasificar a todos los elementos del conjunto de entrenamiento.

1.2. Objetivos

Proponer un procedimiento k-NN donde no sea necesario calcular todas las distancias cada vez que tengamos que predecir un punto nuevo. Dado que solamente podemos calcular un porcentaje de las distancias, las restantes deben ser imputadas, este es el problema al cual nos enfrentamos. Compararemos distintos enfoques que se han propuesto

para completar matrices de distancias [3] [4] [5] en combinación con el procedimiento k-NN en términos del error del vector de distancias y del orden de vecinos resultantes. Los métodos también se compararan por el coste computacional aún sabiendo que esto último depende del ordenador, lenguaje y forma de implementación, aunque para esto siempre se tratará de programar de la manera más eficiente y clara. Todos los experimentos se realizaron usando **R** [6] y todos los códigos desarrollados se pueden encontrar en el siguiente repositorio de GitHub (*https://github.com/aldofranco91/TFM_Ing_Mat*).

1.3. Estructura del trabajo

El resto del trabajo se estructura en los siguientes capítulos:

- Capítulo 2: Se hace una revisión de la literatura sobre algoritmos que nos pueden servir para abordar nuestro problema. Además se plantean aquellas definiciones que nos serán de utilidad para la solución de nuestro problema. Se realizan pruebas mediante simulación para determinar cuál método arroja el mejor resultado.
- Capítulo 3: Se utiliza el método propuesto en el dataset MNIST [7] para un problema de clasificación de múltiples clases (10 clases) y en el dataset DIAMONDS para un problema de regresión.
- Capítulo 4: Se presentan las conclusiones y futuras líneas de trabajo.

2. METODOLOGÍA: ANTECEDENTES Y PROPUESTA

En este capítulo se presentan las definiciones y resultados que nos serán de utilidad para la búsqueda de un método que resuelva nuestro problema junto con una revisión bibliográfica de artículos que abordan problemas similares al nuestro como lo es el problema de la métrica más cercana o el problema de la inferencia filogenética. Se propone una solución basada en procedimientos clúster y se compara el desempeño de las distintas opciones mediante un estudio de simulación.

2.1. Definiciones básicas

Definición 2.1.1. Para un conjunto de elementos X una *distancia* o *métrica* es cualquier función o aplicación $d(a, b) : X \times X \to \mathbb{R}$ que verifique las siguientes condiciones:

- No negatividad: $\forall a, b \in X : d(a, b) \geq 0$.
- Coincidencia: $d(a, b) = 0 \iff a = b$.
- Simetría: $\forall a, b \in X : d(a, b) = d(b, a)$.
- Desigualdad triangular: $\forall a, b, c \in X : d(a, b) \leq d(a, c) + d(c, b)$. [8]

Definición 2.1.2. Sea $M = \left\{m_{ij}\right\}$ una matriz $n \times m$, se dice *no negativa* cuando todas sus entradas son no negativas, es decir $m_{ij} \geq 0 \ \forall i, j$.

Definición 2.1.3. Sea M una matriz $n \times n$, se dice que es *matriz de disimilitud* si es una matriz no negativa, simétrica y con ceros en su diagonal principal.[3]

Definición 2.1.4. Sean x e y dos puntos de $\mathbb{R}^n$ se define la *distancia Manhattan* [9] entre estos puntos como:

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i| . \tag{2.1}$$

Definición 2.1.5. Sean x e y dos puntos de $\mathbb{R}^n$ se define la *distancia euclidiana o euclídea* [9] entre estos puntos como:

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.2}$$

Definición 2.1.6. Sean x e y dos puntos de $\mathbb{R}^n$ se define la *distancia de Chebyshev* [9] entre estos puntos como:

$$d_\infty(x, y) = \max_{i=1,\ldots,n} (|x_i - y_i|). \tag{2.3}$$

Definición 2.1.7. Sean x e y dos puntos de $\mathbb{R}^n$ se define la *distancia de Minkowski* [9] entre estos puntos como:

$$d_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} . \tag{2.4}$$

Notemos que para $p = 1$ tenemos la distancia Manhattan y para $p = 2$ tenemos la distancia euclidiana. Mientras que si hacemos $p \to \infty$ obtenemos la distancia de Chebyshev.

Definición 2.1.8. Sea $M = \left\{m_{ij}\right\}$ una matriz $n \times n$, se dice que es una *matriz de distancias*, si es una matriz de disimilitud cuyas entradas satisfacen la desigualdad triangular. Esto es, M, es matriz de distancias si y solo si para toda terna de índices (i, j, k) se verifica que [10]:

$$m_{ik} \leq m_{ij} + m_{jk}.$$

Definición 2.1.9. Una matriz de distancias, $M = \left\{m_{ij}\right\}$, se dice *aditiva*, cuando sus entradas verifican tanto la desigualdad triangular como la cuadrangular [11]:

$$m_{ij} + m_{kl} \leq \text{máx}[m_{ik} + m_{jl}; m_{il} + m_{jk}] \quad \forall i, j, k, l.$$

Definición 2.1.10. Una matriz de distancias M, se dice *ultramétrica*, cuando es aditiva y sus entradas verifican la desigualdad ultramétrica [12]:

$$m_{ij} \leq \text{máx}[m_{ik}; m_{jk}] \quad \forall\ i, j, k.$$

Definición 2.1.11. Se llama *cardinalidad* al número de elementos de un conjunto A y se denota por $|A|$.[13]

Definición 2.1.12. El *índice o coeficiente de Jaccard* entre dos conjuntos A y B se define como la cardinalidad de la intersección de ambos conjuntos dividida por la cardinalidad de su unión.[14]

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}, \tag{2.5}$$

donde

- $M_{00} + M_{01} + M_{10} + M_{11} = n$.
- M_{11} representa el número total de casos donde A y B tienen un valor de 1.
- M_{01} representa el número total de casos donde el valor de A es 0 y el valor de B es 1.
- M_{10} representa el número total de casos donde el valor de A es 1 y el valor de B es 0.
- M_{00} representa el número total de casos donde A y B tienen un valor de 0.

		A	
		0	1
B	0	M_{00}	M_{10}
	1	M_{01}	M_{11}

El índice de Jaccard siempre toma valores entre 0 y 1, correspondiendo este último a la igualdad entre ambos conjuntos.

Es importante destacar que existe otra medida que se llama el *coeficiente de coincidencia simple* (SMC) [15] o *coeficiente de similitud de Rand* que es muy similar al Jaccard pues se define como:

$$SMC = \frac{M_{00} + M_{11}}{M_{01} + M_{10} + M_{11}}.$$

Definición 2.1.13. Se define el *error absoluto medio* (MAE), como:

$$MAE = \frac{\sum_{i=1}^{k} \left|\widehat{O_i} - O_i\right|}{k},$$

donde $\widehat{O_i}$ son los valores estimados y O_i los valores reales u observados, siendo k la cantidad de valores a estimar.[16]

Definición 2.1.14. Sea O la matriz original y P la matriz pronosticada, se define la *diferencia relativa entre estas matrices* como:

$$\frac{\|O - P\|_2}{\|O\|_2},$$

donde $\|A\|_2 = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \left|a_{ij}\right|^2}$.

Definición 2.1.15. Sea o el vector original y p el vector pronosticado, se define la *diferencia relativa entre estos vectores* como:

$$\frac{\|o - p\|_2}{\|o\|_2},$$

donde $\|o\|_2 = \sqrt{\sum_{i=1}^{n} o_i^2}$.

2.2. El problema de la métrica más cercana

Supongamos que tenemos una matriz D de dimensión $n \times n$ cuyos elementos deberían cumplir las desigualdades triangulares pero, por errores de medición o incluso por omisión de algunas mediciones, estas desigualdades no se verifican. El *problema de la métrica más cercana* consiste en encontrar una matriz M cuyos elementos cumplan las correspondientes desigualdades triangulares y que M esté próxima a la matriz D. Este problema ha sido estudiado en [3] [10] y puede formularse como sigue:

> Si hay n puntos, podemos recoger las medidas en una matriz simétrica D de $n \times n$ cuya entrada (j, k) representa la "distancia" entre los objetos j y k. Luego, buscamos aproximar esta matriz por otra matriz M cuyas entradas satisfacen las desigualdades triangulares. Es decir, $m_{ik} \leq m_{ij} + m_{jk}$ por cada tupla (i, j, k). En otras palabras se requiere una matriz de distancias M que sea la más cercana a una matriz de disimilitud dada D con respecto a alguna norma entre matrices. Específicamente, se busca una matriz de distancias M tal que,
>
> $$M \in \left\{argmin_{X \in M_N} \left\|W \odot (X - D)\right\|\right\}, \tag{2.6}$$
>
> donde M_N es el conjunto de todas las matrices de distancias $n \times n$, este conjunto es cerrado y convexo. Por otra parte W es una matriz $n \times n$ de pesos, simétrica y no negativa y $\odot$ denota la multiplicación elemento a elemento entre dos matrices. La matriz de pesos W refleja nuestra confianza en las entradas de la matriz D. Por ejemplo, cuando cada d_{ij} representa una medida

con varianza σ_{ij}^2, podríamos establecer $w_{ij} = 1/\sigma_{ij}^2$ y si falta una entrada en D, se pone a cero el peso correspondiente. Generalmente se asume que todos los pesos son iguales a uno. En [17] se proponen algoritmos para resolver el problema (2.6) para las normas L_1, L_2 y L_∞.

Teorema 2.2.1. La función $\|W \odot (X - D)\|$ siempre alcanza su mínimo en M_N. Además, cada mínimo local es un mínimo global. Si, además, la norma es estrictamente convexa y la matriz de pesos no tiene ceros o infinitos fuera de su diagonal, entonces hay un mínimo global único.

La demostración de este teorema se encuentra en [3] haciendo uso de resultados expuestos en [18].

En principio, es posible utilizar cualquier norma entre matrices para este problema, pero lo analizaremos para las normas l_p. Con lo cual los problemas asociados son:

$$\min_{X \in M_N} \left[\sum_{j \neq j} \left| w_{jk}(x_{jk} - d_{jk}) \right|^p \right]^{1/p} \qquad 1 \leq p < \infty \tag{2.7}$$

y

$$\min_{X \in M_N} \max_{j \neq k} \left| w_{jk}(x_{jk} - d_{jk}) \right| \quad p = \infty \tag{2.8}$$

Para resolver los problemas (2.7) y (2.8) puede parecer que se debería utilizar programación lineal (LP) cuando $p = 1, \infty$, y programación convexa para $p \neq 1, \infty$, pero resulta que los requisitos de tiempo y almacenamiento de estos enfoques son prohibitivos.[17]

Siguiendo [17], se planteará el algoritmo para $p = 2$ puesto que este caso resulta ser el más simple y juega un papel fundamental en la resolución de los problemas para $p = 1$ y $p = \infty$:

Dado un vector de disimilitud d, deseamos encontrar su proyección ortogonal m sobre el cono M_n. Definamos $e = m - d$ que representaría los cambios a las distancias originales y $b = -Ad$ que indica en qué medida se viola la desigualdad triangular. El problema (2.7) se convierte en

$$\begin{aligned} &\text{mín } \|e\|_2 , \\ &\text{sujeto a: } Ae \leq b. \end{aligned} \tag{2.9}$$

A partir del minimizador e^*, usamos la ecuación $m^* = d + e^*$ para recuperar el vector con la distancia óptima. Iniciamos el vector en $e = 0$, y empezamos a recorrer todas las tuplas (i, j, k). Supongamos que se incumple la desigualdad triangular, con lo cual $e_{ij} - e_{jk} - e_{ki} > b_{ijk}$. Deseamos remediar este incumplimiento haciendo un ajuste l_2 de e_{ij}, e_{jk} y e_{ki}. En otras palabras, el vector e se proyecta ortogonalmente sobre el conjunto de restricciones $\left\{ x : x_{ij} - x_{jk} - x_{ki} \leq b_{ijk} \right\}$. Esto equivale a resolver:

$$\begin{aligned} &\text{mín } \frac{1}{2} \left[(x_{ij} - e_{ij})^2 + (x_{jk} - e_{jk})^2 + (x_{ki} - e_{ki})^2 \right], \\ &\text{sujeto a: } x_{ij} - x_{jk} - x_{ki} = b. \end{aligned} \tag{2.10}$$

Sin embargo se verifica fácilmente que la solución del problema anterior está dada por:

$$\begin{aligned} x_{ij} &\leftarrow e_{ij} + \mu, \\ x_{jk} &\leftarrow e_{jk} + \mu, \\ x_{ki} &\leftarrow e_{ki} + \mu, \end{aligned} \tag{2.11}$$

con $\mu = \dfrac{-b + e_{ij} - e_{jk} - e_{ki}}{3}$. Por lo tanto, solo tres componentes del vector e tienen que ser actualizados, y sus valores están dados por (2.11).

A su vez, fijamos la desigualdad de cada tupla donde no se cumpla la desigualdad triangular usando (2.11). También debemos introducir un término de corrección para guiar el algoritmo al mínimo global. Las correcciones tienen una interpretación simple en términos de la dualidad del problema de la minimización (2.9): cada variable dual corresponde al incumplimiento en una sola desigualdad triangular, y cada corrección individual resulta en una disminución de dicho incumplimiento. El procedimiento se repite hasta que ninguna tupla reciba una actualización significativa.

El Algoritmo 1 muestra el esquema iterativo completo.

Algoritmo 1: Algoritmo Triangle Fixing para l_2.

Entrada: D: Matriz de disimilitud y ϵ la tolerancia.
Salida : $M = argmin_{X \in M_N} \left\|(X - D)\right\|_2$
for $1 \leq i < j < k \leq n$ **do**
  $z_{ijk} \leftarrow 0$
end
for $1 \leq i < j \leq n$ **do**
  $e_{ij} \leftarrow 0$
end
$\delta \leftarrow 1 + \epsilon$
while $\delta > \epsilon$ **do**
  foreach *Incumplimiento de la desigualdad triangular para* (i, j, k) **do**
    $b \leftarrow d_{ki} + d_{jk} + d_{ij}$
    $\mu \leftarrow -\dfrac{1}{3}(b - e_{ij} - e_{jk} - e_{ki})$
    $\theta \leftarrow \text{mín}\left\{\mu, z_{ijk}\right\}$
    $e_{ij} \leftarrow e_{ij} + \theta$
    $e_{jk} \leftarrow e_{jk} - \theta$
    $e_{ki} \leftarrow e_{ki} - \theta$
    $z_{ijk} \leftarrow z_{ijk} - \theta$
  end
  $\theta \leftarrow$ *Suma de todos los cambios en los valores de e.*
end
return $M = D + E$

En nuestras simulaciones utilizaremos $p = 2$. El algoritmo para este valor está implementado en **R** en la librería *dtw* (Dynamic Time Warping) [19]. Según comunicación con el autor de dicho paquete estos algoritmos pueden fallar, sobre todo cuando n es muy

grande. En nuestro caso fallaba a partir de $n = 800$.

Llegados a este punto, nos preguntamos ¿qué relación tiene el problema de la métrica más cercana con nuestro problema? Si asumimos que conocemos todas las distancias entre todos los n objetos en nuestra muestra de entrenamiento podemos crear una nueva matriz $(n + 1) \times (n + 1)$ donde la columna/fila $(n + 1)$ serían las distancias del punto x_0 a las restantes teniendo en cuenta que tendremos un $l\,\%$ de distancias sin calcular y podemos utilizar el Algoritmo 1 para imputar esos valores. A estos valores faltantes tenemos que darle algún valor inicial pues el algoritmo que se usa para resolver el problema de la métrica más cercana necesita valores numéricos.

Los primeros enfoques para imputar estos valores faltantes fueron:

- $d(x_0, x_j) = 0$, siendo j aquel objeto del cual no tenemos la distancia real. Este primer enfoque fue un total fracaso con lo cual no reportamos resultados pues en el tiempo de ejecución era extremadamente alto al igual que las diferencias relativa entre matrices y vectores.
- $d(x_0, x_j) = \dfrac{\sum_{i=1}^{n} d(x_i, x_j)}{n} \quad , i \neq j.$
- $d(x_0, x_j) = \text{mediana}(d(x_i, x_j)) \quad , \forall i \neq j.$

Para comparar las distintas propuestas de valores iniciales de las distancias desconocidas realizamos un estudio de simulación que consta de los siguientes pasos:

1. Se generan $n = 800$ puntos de una distribución normal multivariante con $\mu = \vec{0}_p$ y $\sigma = I_p$, siendo $p = 20$.
2. Se calcula la matriz de distancias, usando la distancia euclidiana.
3. Se genera un nuevo punto x_0 y se calculan las n distancias.
4. Se asumen conocidas las distancias de x_0 a $n(1 - l) = 80$ puntos, $(l = 0.9)$ y las restantes distancias se imputan usando los métodos descritos anteriormente, con lo cual tenemos una matriz de disimilitud D.
5. Se aplica el triangle fixing a D y nos devuelve una matriz M.
6. Se calcula la diferencia relativa entre la matriz M y la matriz de distancias original al igual que la diferencia relativa entre el vector de distancias correspondientes a x_0 en la matriz original y en M. También se registra el tiempo de cómputo.
7. Todo lo anterior se hace $N = 2000$ veces.

Se puede apreciar en la figura 2.1 las diferencias relativas entre las matrices que devuelve el algoritmo y las matrices originales, al igual que la diferencias relativas entre el vector que contiene las distancias originales y el que devuelve el algoritmo, ver figura 2.2.

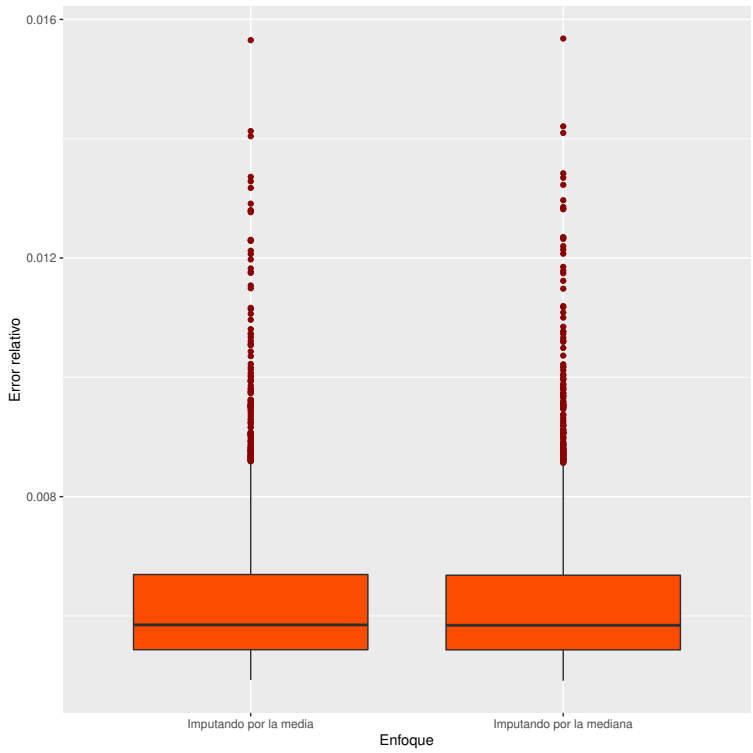

Figura 2.1: Diagramas de caja de las diferencias relativas entre la matriz de distancias original y la matriz imputada M.

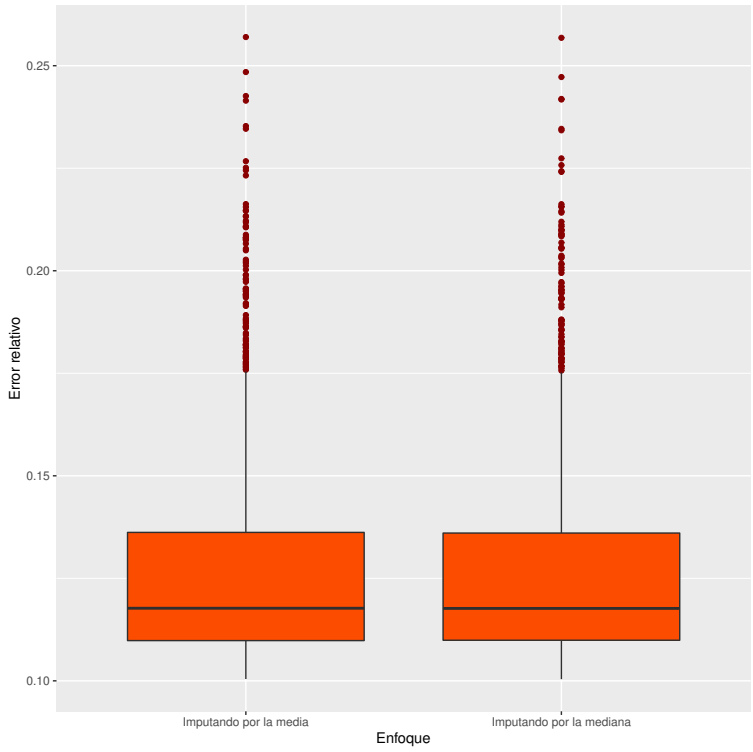

Figura 2.2: Diagramas de caja de las diferencias relativas entre las distancias reales e imputadas.

Estas grandes diferencias y altos costes en tiempo (como se puede apreciar en la fi-

gura 2.3) viene dado por el hecho que nuestras imputaciones no verifican holgadamente las desigualdades triangulares, con lo cual el algoritmo tiene que iterar muchas veces. Por tanto debemos buscar una forma de mejorar estas aproximaciones iniciales.

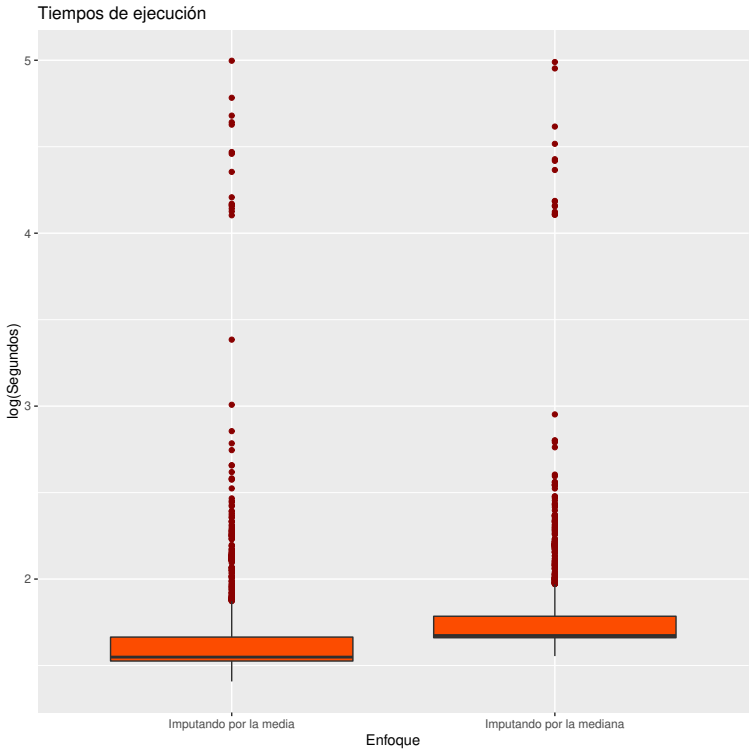

Figura 2.3: Diagramas de caja de los tiempos de ejecución para cada enfoque en escala logarítmica.

2.2.1. Método propuesto

De acuerdo a las restricciones impuestas a nuestro problema, sabemos que:

- Las distancias $d(x_i, x_j)$ son conocidas $\forall i, j = \{1, ..., n\}$.
- Podemos calcular las distancias de $d(x_0, x_i)$ cuando $i \in I$ y $|I| << n$.
- No podemos calcular las $d(x_0, x_i)$ cuando $i \in I^c$.
- $|I| \approx (1 - l)n$ y $|I^c| \approx ln$.

Por otra parte haciendo uso de la desigualdad triangular tenemos:

$$d(x_0, x_{i^*}) \leq d(x_0, x_i) + d(x_i, x_{i^*}) \quad \forall i^* \in I^c, \tag{2.12}$$

por tanto obtenemos que:

$$d(x_0, x_{i^*}) \leq \min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i^*})\}. \tag{2.13}$$

De igual forma, tenemos que:

$$d(x_i, x_{i^*}) \leq d(x_0, x_{i^*}) + d(x_0, x_i) \tag{2.14}$$

y

$$d(x_0, x_i) \leq d(x_0, x_{i^*}) + d(x_i, x_{i^*}), \tag{2.15}$$

por tanto, obtenemos que:

$$\max_{i \in I} |d(x_0, x_i) - d(x_i, x_{i^*})| \leq d(x_0, x_{i^*}). \tag{2.16}$$

Finalmente,

$$\max_{i \in I} |d(x_0, x_i) - d(x_i, x_{i^*})| \leq d(x_0, x_{i^*}) \leq \min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i^*})\} \tag{2.17}$$

Como ya tenemos acotada $d(x_0, x_{i^*})$, asignaremos a dicha distancia la media de sus cotas:

$$d(x_0, x_{i^*}) = \frac{1}{2}\left(\min_{i \in I} \{d(x_0, x_i) + d(x_i, x_{i^*})\} + \max_{i \in I} |d(x_0, x_i) - d(x_i, x_{i^*})|\right). \tag{2.18}$$

Cuando asignamos los valores imputados usando la expresión (2.18) para posteriormente aplicar el triangle fixing obtenemos que la matriz D coincide con la matriz M que devuelve dicho algoritmo. Con lo cual parece ser que esos valores imputados satisfacen todas las condiciones que debe verificar una matriz de distancia. Esto solo se ha comprobado mediante simulación utilizando $N = 10000$ y valores de l entre 0.6 y 0.9, pero no tenemos una prueba formal de esta conjetura.

Dado el coste computacional del algoritmo triangle fixing directamente imputaremos usando la expresión (2.18). Hasta ahora hemos seleccionados puntos al azar, en lo que sigue propondremos un método que nos ayude a mejorar estos resultados basándonos en métodos clúster.

2.3. Clúster o agrupamiento

El clustering o agrupamiento es una técnica de aprendizaje no supervisada que intenta encontrar relaciones entre variables pero no la relación que guardan con respecto a una variable objetivo, es decir sin hacer uso de las etiquetas. El conjunto de datos tiene que ser dividido automáticamente en clústeres, de manera que los objetos dentro del mismo clúster sean similares, mientras que los objetos de diferentes clústeres sean menos semejantes. No existe una definición general de clúster, lo que significa que diferentes enfoques pueden obtener diferentes clústeres del mismo conjunto de datos.

El motivo por el cual el agrupamiento será de gran importancia para nosotros está dado por el hecho de que hasta ahora hemos seleccionado los puntos a los cuales se le calculan las distancias de una forma totalmente aleatoria. Pero usaremos, la idea propuesta en [20], donde en un conjunto de datos muy grandes se decide aplicar k-NN por submuestras y cada submuestra es definida mediante un clúster. A continuación se puede ver un resumen de dicho algoritmo.

Algoritmo 2: Algoritmo k-NN para Big Data.

Entrada: Datos de entrenamiento y dato a clasificar x_0.
Salida : Etiqueta.
1. Producir K clústeres, $C_1, C_2, ..., C_K$.
2. Calcular la distancia de x_0 a todos los centroides, $d(x_0, C_i)\ \forall i = 1, ..., K$.
3. Buscar el centroide(C_i) más cercano a x_0.
4. Crear un nuevo conjunto de datos, correspondientes a los puntos que pertenecen a C_i.
5. Usar estos datos como datos de entrenamiento para predecir x_0.

Existen muchos métodos para agrupar como la agrupación jerárquica, los métodos k-medias, PAM (Partitioning Around Medoids), y DBSCAN (Density-Based Spatial Clustering of Application with Noise). Estos son de los más conocidos y usados [21], sin embargo dado el impacto del Big Data existen trabajos que intentan paralelizar estos procedimientos y escalar a conjuntos de datos más grandes.

2.3.1. Agrupamiento jerárquico

El agrupamiento jerárquico es una técnica de aprendizaje no supervisada, la cual busca construir una jerarquía de grupos. Las estrategias para agrupamiento jerárquico generalmente se dividen en dos tipos:

- Aglomerativas: Este es un enfoque ascendente, cada observación comienza en su propio grupo, y los pares de grupos son mezclados mientras se sube en la jerarquía.
- Divisivas: Todas las observaciones comienzan en un grupo, y se realizan divisiones mientras se baja en la jerarquía.

Los resultados del agrupamiento jerárquico son muchas veces presentados mediante un dendrograma que no es más que una representación gráfica en forma de árbol. En el caso general, la complejidad del agrupamiento aglomerativo es $O(n^3)$ lo cual es muy ineficiente cuando n es grande, sin embargo el agrupamiento divisivo con búsqueda exhaustiva es $O(2^n)$ lo cual es mucho peor.

Para decidir qué grupos deberían ser combinados (tipo aglomerativo), o cuando un grupo debería ser dividido (tipo divisivo), es necesario una medida de disimilitud entre los conjuntos. Es importante recalcar que la elección de una métrica apropiada influirá en la forma de los grupos, ya que algunos conjuntos pueden estar cerca unos de otros de acuerdo a una distancia y más lejos de acuerdo a otra. En el cuadro 2.1, presentamos las tres métricas más conocidas. [22]

Cuadro 2.1: Criterios de agrupamiento jerárquico.

Nombre	Fórmula
Agrupamiento de enlace máximo o completo.	$\max \{d(x, y) : x \in X, y \in Y\}$
Agrupamiento de enlace mínimo o simple.	$\min \{d(x, y) : x \in X, y \in Y\}$
Agrupamiento de enlace promedio, o UPGMA.	$\frac{1}{\lvert X\rvert \lvert Y\rvert} \sum_{x \in X} \sum_{y \in Y} d(x, y)$

A continuación se puede apreciar cómo diferentes tipos de enlaces nos devuelve diferentes resultados (se usó un punto de corte $K = 5$), los conjuntos de datos fueron descargados de [23] y se conocen como conjuntos de datos de juguete.

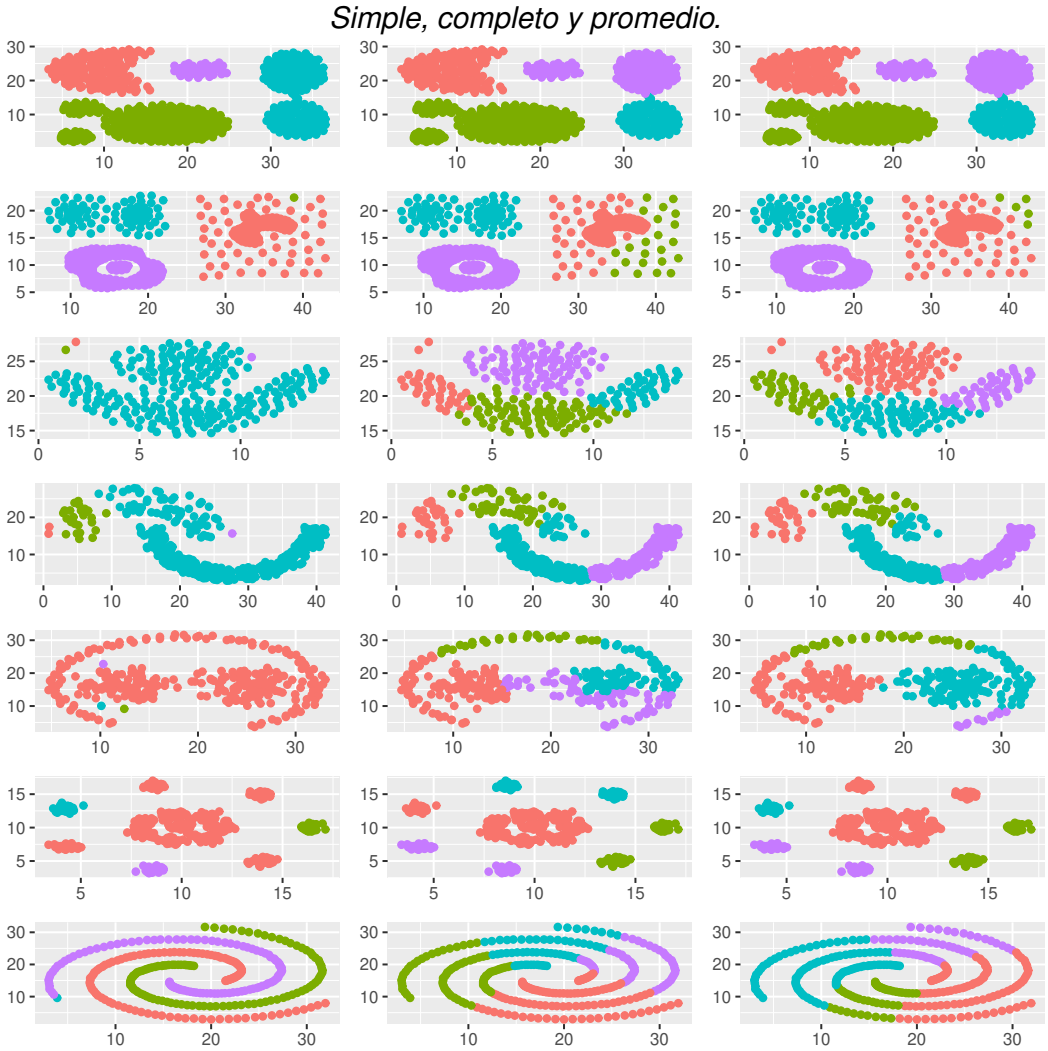

Figura 2.4: Comparación de diferentes métodos de agrupamiento jerárquico.

2.3.2. K-medias

Otro de los algoritmos más usados es el K-medias, el cual tiene como objetivo crear K grupos a partir de n observaciones, en el que cada una pertenece a un grupo cuyo valor medio es el más cercano. Sean $(x_1, ..., x_n)$ las observaciones, el algoritmo construye una partición de dichas observaciones, $C = \{C_1, ..., C_K\}$, con el fin de minimizar la suma de los cuadrados dentro de cada grupo:

$$\arg\min_{C} \sum_{i=1}^{K} \sum_{x_j \in C_i} \left\| x_j - \mu_i \right\|_2^2, \tag{2.19}$$

donde μ_i es la media de C_i. Los $\{\mu_i\}$ se les llama centroides.

Desafortunadamente, aunque el agrupamiento de K-medias es bastante eficiente en tiempo computacional se sabe que es sensible a los valores atípicos. Por esta razón, a

veces se utiliza la agrupación de K-medoids, donde se consideran objetos representativos dentro de cada clúster en lugar de los centroides.

2.3.3. K-medoides

K-medoides es una familia de algoritmos que escogen puntos del conjunto de datos como centros y trabaja con una métrica arbitraria de distancias. Es más robusto ante atípicos que K-medias porque minimiza una suma de disimilaridades (entre pares de puntos) en vez de una suma de distancias euclidianas cuadradas. Un medoide puede ser definido, como el objeto de un grupo cuya disimilaridad media a todos los objetos en el grupo es mínima.

La implementación práctica más común de K-medoides es el algoritmo Partición Alrededor de Medoides (PAM) y sea plantea como sigue:

Algoritmo 3: PAM (Partición Alrededor de Medoides)

Entrada: Seleccionar K de los n puntos como medoides.
Salida : K medoides
Asociar cada punto al medoide más cercano.
while *Coste de la configuración disminuya* **do**
  foreach *medoide* $\underline{m}$ *y medoide* $\underline{o}$ **do**
    1. Intercambiar $\underline{m}$ y $\underline{o}$, recalcular el coste (suma de la distancia de los puntos a sus medoids).
    2. Si el coste total de la configuración aumentó en el paso anterior, deshacer el intercambio.
  end
end

Entre los algoritmos K-medoides que existen se encuentra el antes mencionado PAM que se considera uno de los más potentes, el CLARA y CLARANS. Sin embargo, PAM tiene el inconveniente de que funciona de manera ineficiente para conjuntos de datos grande (que es nuestro caso).[24]

Sin embargo existe una versión basado en K-medias para los medioides, este algoritmo se llama $fastkmed$ y está implementado en la librería *kmed* [25]. Además requiere la distancia entre cada par de objetos una sola vez y el resultado de varias simulaciones muestran que dicho método tiene mejor rendimiento que el agrupamiento K-medias. También se reduce significativamente el tiempo de ejecución con respecto al PAM con un rendimiento comparable, PAM es del orden de $O(k(n-k)^2)$ y el $fastkmed$ del orden de $O(nk)$ muy similar al K-medias. [26]

El algoritmo $fastkmed$ se define como sigue:

Algoritmo 4: Algoritmo fastkmed.

Entrada: Seleccionar K de los n puntos como medoides.
Salida : K medoides.

1. Seleccione los medoides iniciales:

   - Calcular la distancia entre cada par de objetos.
   - Calcular v_j para el objeto j como sigue:

$$v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}}, \ j = 1, ..., n$$

   - Ordenar v_j en orden ascendente. Seleccionar k objetos teniendo los primeros k valores más pequeños como medoides iniciales.
   - Obtener el resultado del clúster inicial asignando cada objeto al medoide más cercano.
   - Calcular la suma de distancias de todos los objetos a sus medoides.

2. Actualizar medoides:

   - Encontrar un nuevo medoide de cada grupo, que es el objeto que minimiza la distancia total a otro objeto en su clúster. Actualizar el medoide actual en cada grupo reemplazando con el nuevo medoide.

3. Asignar objetos a los medoides:

   - Asignar cada objeto al medoide más cercano y obtener el resultado del clúster.
   - Calcular la suma de la distancia de todos los objetos a sus medoides. Si la suma es igual a la anterior, detener el algoritmo y en caso contrario, volver al Paso 2.

El algoritmo tiene una característica excelente, requiere que la distancia entre cada par de objetos se use una sola vez. Según los autores, el resultado de varias simulaciones que utilizan conjuntos de datos artificiales muestra que el *fastkmed* tiene mejor rendimiento que el agrupamiento K-medias y reduce significativamente el tiempo de cálculo con respecto a PAM incluso con un rendimiento mejor. También la selección inicial de medoides empleado en dicho método funciona bastante bien en comparación con otros métodos.[26]

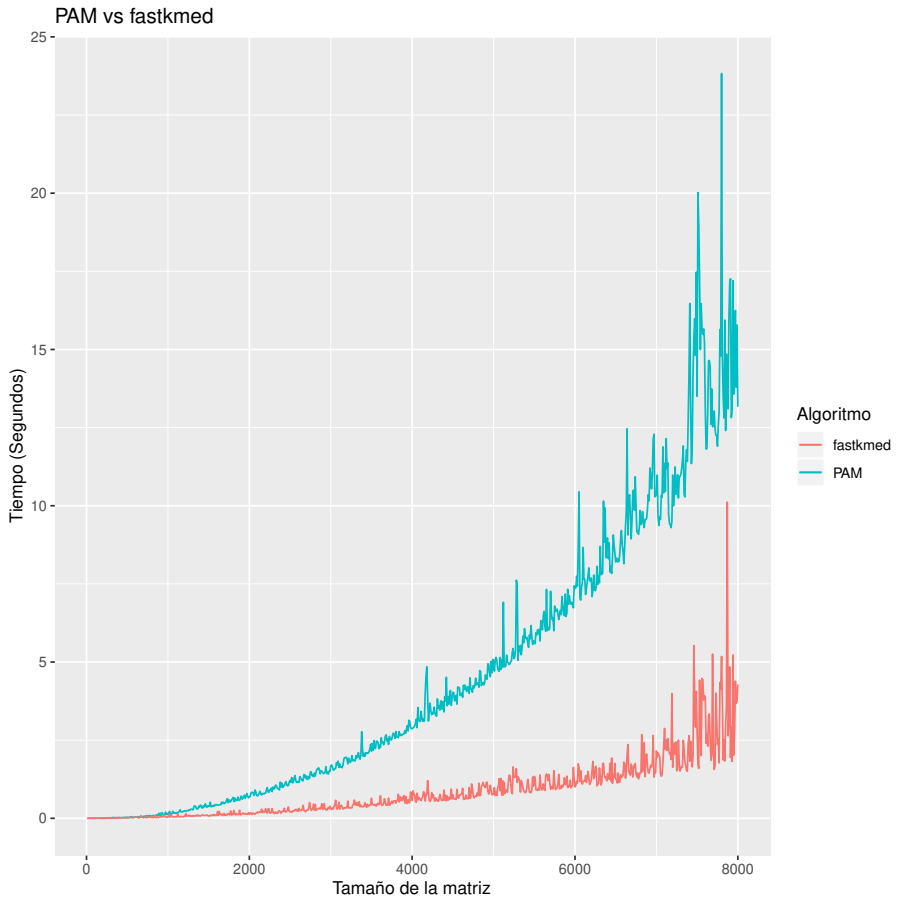

Figura 2.5: Comparación del PAM vs $fastkmed$ para $K = 3$.

En la figura 2.5 mostramos el tiempo requerido por PAM y $fastkmed$. En la comparación el PAM se está compilando en su versión más rápida entre todas las propuestas [27] pues tiene diferentes versiones las cuales:

- Reducen el tiempo de ejecución en un factor de $O(k)$.
- Permiten ejecutar múltiples intercambios por iteración, generalmente reduciendo el número de iteraciones.
- Permiten agregar optimizaciones menores y se espera que sea la variante más rápida.

Todas estas propuestas están implementadas para **R** haciendo uso de la librería *cluster* [28].

2.4. Aditivo y ultramétrico

El problema de la inferencia filogenética a partir de conjuntos de datos que incluyen entradas incompletas o inciertas es uno de los temas más relevantes en biología sistemática.[29] El objetivo es inferir filogenias a partir de datos evolutivos, incluida información faltante o incierta, por ejemplo, cuando las secuencias de nucleótidos o proteínas observadas contienen huecos o entradas faltantes.

Entre los diferentes algoritmos que existen sobre inferencia filogenética hay dos que se han usado para completar matrices de distancias D. Estos métodos han sido propuestos por [4] y [5] para ultramétricas, mientras que para matrices de distancias aditivas por [30] y [31]. Ambos algoritmos están implementados en **R**, en la librería *ape* (Analyses of Phylogenetics and Evolution) [32].

Algoritmo 5: Algoritmo para distancias ultramétricas.

Entrada: Distancia parcial d en el conjunto de n taxones.
Salida : Distancia completa o parcial d en el conjunto de n taxones.
1. Contar el número de NAs en d, se define como c.
2. **foreach** $d(i, j)$ *que sea NA de* d **do**
   $MinMax$ es la máxima entrada de d.
   foreach k *tal que* $d(i,k)$ *y* $d(j,k)$ *son valores conocidos de* d **do**
      $Max = \max(d(i,k); d(j,k))$
      if $Max < MinMax$ **then**
         $Max = MinMax$
      end
   end
   if *Si hay al menos un par conocido de entradas* $d(i,k)$ *y* $d(j,k)$ **then**
      $d(i, j) = MinMax$
      $c = c - 1$
   end
end
3. Si ha cambiado c, ir al punto 2.

Algoritmo 6: Algoritmo para distancias aditivas.

Entrada: Distancia parcial d en el conjunto de n taxones.
Salida : Distancia completa o parcial d en el conjunto de n taxones.
1. Contar el número de NAs en d, se define como c.
2. **foreach** $d(i, j)$ *que sea NA de* d **do**
   $MinMax$ es la máxima entrada de d.
   foreach k, l *tal que* $d(i,k)$,$d(j,k)$, $d(i,l)$,$d(j,l)$ *y* $d(k,l)$ *son valores conocidos de* d **do**
      $Max = \max(d(i,k) + d(j,l); d(i,l) + d(j,k) - d(k,l))$
      if $Max < MinMax$ **then**
         $Max = MinMax$
      end
   end
   if *Si al menos las entradas* $d(i,k)$,$d(j,k)$, $d(i,l)$,$d(j,l)$ *y* $d(k,l)$ *son conocidas* **then**
      $d(i, j) = MinMax$
      $c = c - 1$
   end
end
3. Si ha cambiado c, ir al punto 2.

Estos algoritmos serán utilizados pues no requieren ningún tipo de imputación inicial. De hecho los valores faltantes deben definirse como NA para poder hacer uso de estos

métodos.

2.5. Estudio de simulación

En esta sección se compararán los diferentes métodos propuestos y para eso usaremos dos métricas, el MAE y el índice de Jaccard. Haremos simulaciones donde compararemos los resultados observación a observación, es decir en ningún momento analizaremos las etiquetas que pueden tener estas observaciones. Este ejercicio se hará en el próximo capítulo para los conjuntos de datos MNIST y DIAMONDS usando la mejor estrategia que salga de estas simulaciones.

El motivo por el cual usaremos ambas métricas viene dado por el hecho de que el índice de Jaccard presenta un inconveniente. Por ejemplo, supongamos que los vecinos más cercanos para una observación de validación x_0 son las observaciones $A = \{10, 15, 20, 30\}$ de la muestra de entrenamiento y obtenemos que un algoritmo nos devuelve los siguientes puntos más cercanos $B_1 = \{10, 15, 20, 35\}$ y otro algoritmo los puntos $B_2 = \{10, 15, 20, 40\}$ con lo cual se obtiene que

$$J(A, B_1) = \frac{|A \cap B_1|}{|A \cup B_1|} = \frac{3}{5} = \frac{|A \cap B_2|}{|A \cup B_2|} = J(A, B_2). \tag{2.20}$$

Vemos que el índice de Jaccard coincide y no permite diferenciar entre ambos resultados. Con el MAE seleccionaríamos el resultado con menor error de "imputación" de las distancias.

Primero estudiaremos el número de clústeres que debemos escoger de tal manera que el índice de Jaccard sea mayor. Es importante destacar que este problema es diferente al que resuelven métodos como el del codo (Elbow method), brecha (Gap) o el de la silueta (Silhoutte Method) sobre el número de clústeres presentes en un conjunto de datos.

Para encontrar un número "óptimo" de clústeres mediante simulación se hizo lo siguiente:

1. Se generan $n = 3000$ puntos de una distribución normal multivariante con $\mu = \vec{0}_p$ y $\sigma = I_p$, siendo $p = 50$.

2. Se calcula la matriz de distancias, usando la distancia euclidiana.

3. Se aplica $fastkmed$ a dicha matriz de distancias para diferentes valores de $K = (2, 4, 8, 16, 32, 64, 150, 300)$, aquí el valor de K es la cantidad de clústeres, de tal manera que se obtienen K mediodes $\{C_1, ..., C_K\}$.

4. Se genera un nuevo punto x_0.

5. Calcular las distancias $d(x_0, C_i)$ con $i = 1, ..., K$ y se ordenan de menor a mayor.

6. En este punto ya hemos calculado K distancias con lo cual las restantes se harán seleccionando de cada clúster más cercano hasta que se hayan calculado el número máximo de distancias $n(1 - l) = 300$, pues $l = 0.9$.

7. Para los diferentes valores de K expuestos se calcula el índice de Jaccard y el MAE entre el conjunto real de puntos más cercano y el conjunto de puntos más cercano que se obtiene imputando, para $k = 15$ vecinos.

8. Se repiten los pasos $4-7$, $N = 200$ veces.

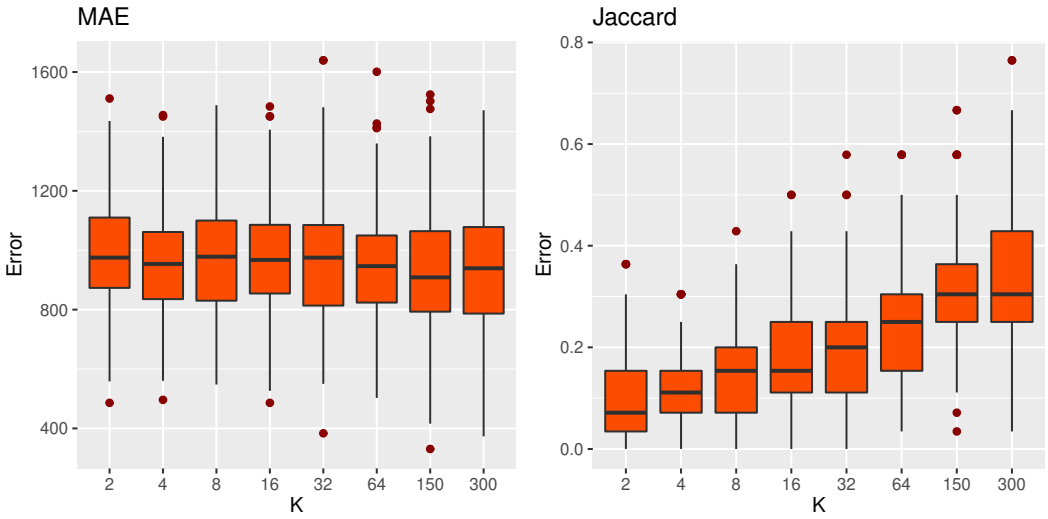

Figura 2.6: Búsqueda de un valor "óptimo" de clústeres.

Se puede observar en la figura 2.6 que a medida que el número de clústeres aumenta el índice de Jaccard mejora, con lo cual a partir de este momento decidiremos usar como número de clústeres $K = 150$ que es el valor correspondiente a $K = (n - ln)/2$, pues presenta menor variabilidad que $n - ln$ que sería el valor correspondiente a $K = 300$. También observamos que el MAE decrece lentamente con el número de clústeres.

Se sabe que en k-NN elegir el valor del k, a veces, resulta ser una tarea nada fácil, con lo cual se decide usar desde $k = 1$ hasta $k = \sqrt{n}$, usando solo los k que tomen valores impares [33]. Las siguientes comparaciones consideran estos valores de k para evaluar el desempeño de las distintas propuestas en función de este parámetro esencial del procedimiento k-NN.

A continuación veremos si hay alguna mejora cuando imputamos usando puntos al azar o usando la idea del clúster, las simulaciones se hicieron de la siguiente forma:

1. Se generan $n = 5000$ puntos de una distribución normal multivariante con $\mu = \vec{0}_p$ y $\sigma = I_p$, siendo $p = 50$.

2. Se calcula la matriz de distancias, usando la distancia euclidiana.

3. Se genera un nuevo punto x_0.

4. Se calculan las distancias de x_0 a los n puntos.

5. Se calculan las distancias de x_0 a $n(1 - l)$ puntos al azar y también a la misma cantidad usando la idea del clúster antes mencionada con $K = (n - ln)/2$. En ambos casos las restantes distancias se imputan usando la expresión (2.18).

6. Se ordenan dichas distancias y se extrae cuáles son los k puntos más cercanos, el máximo valor que toma k es $\sqrt{n}/2$ dado que n es muy grande.

7. Para diferentes valores de k se calcula el índice de Jaccard y el MAE entre el conjunto de puntos más cercano real y el conjunto de puntos más cercano que se obtiene imputando usando ambos métodos.

8. Todo lo anterior se hace $N = 1000$ veces.

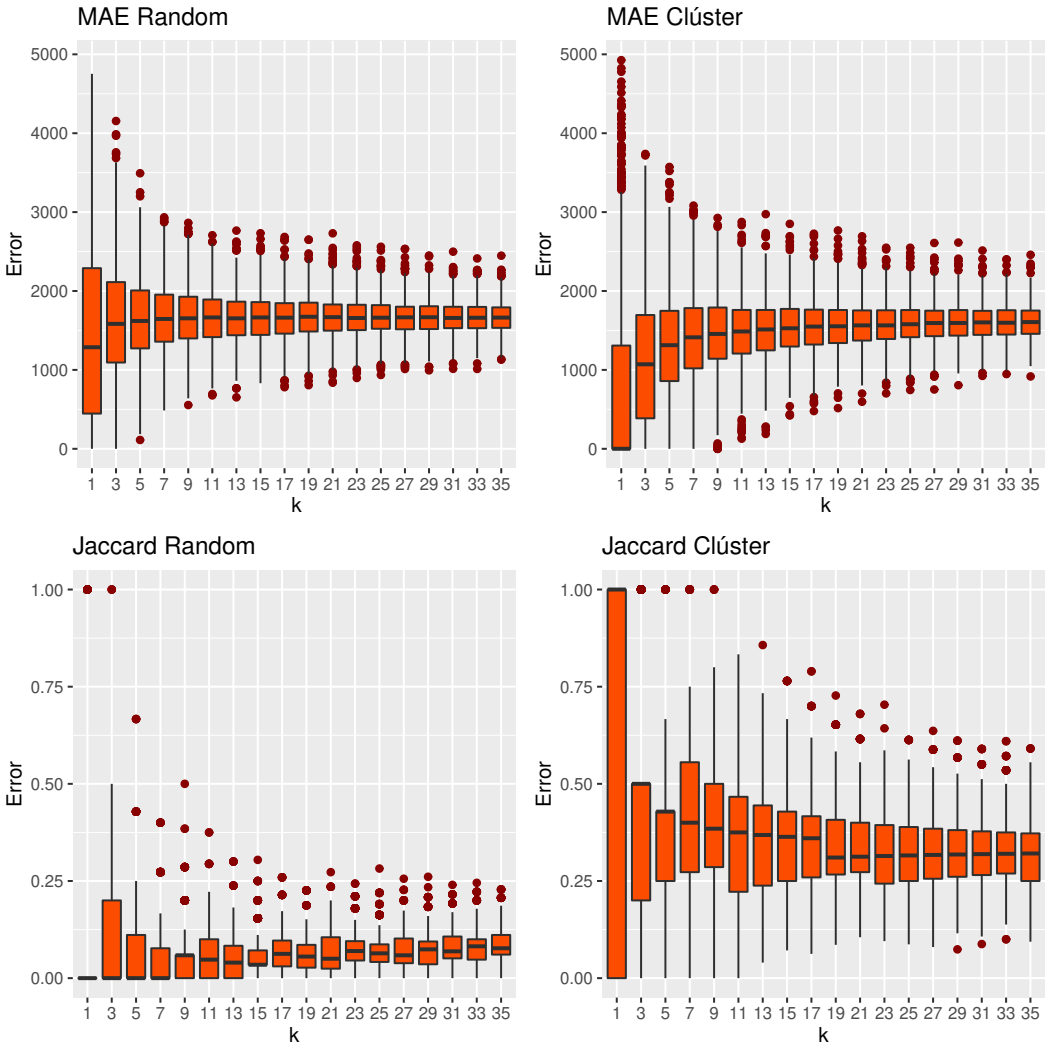

Figura 2.7: Simulación comparando puntos al azar vs. puntos vía clúster.

En la figura 2.7 se puede apreciar claramente que calcular distancias al azar tiene un desempeño inferior al procedimiento basado en clústeres tanto para el MAE como para el índice de Jaccard. También hemos realizado la comparación entre los métodos ultramétrico y aditivo, viendo claramente que el ultramétrico es mejor, ver figura 2.8.

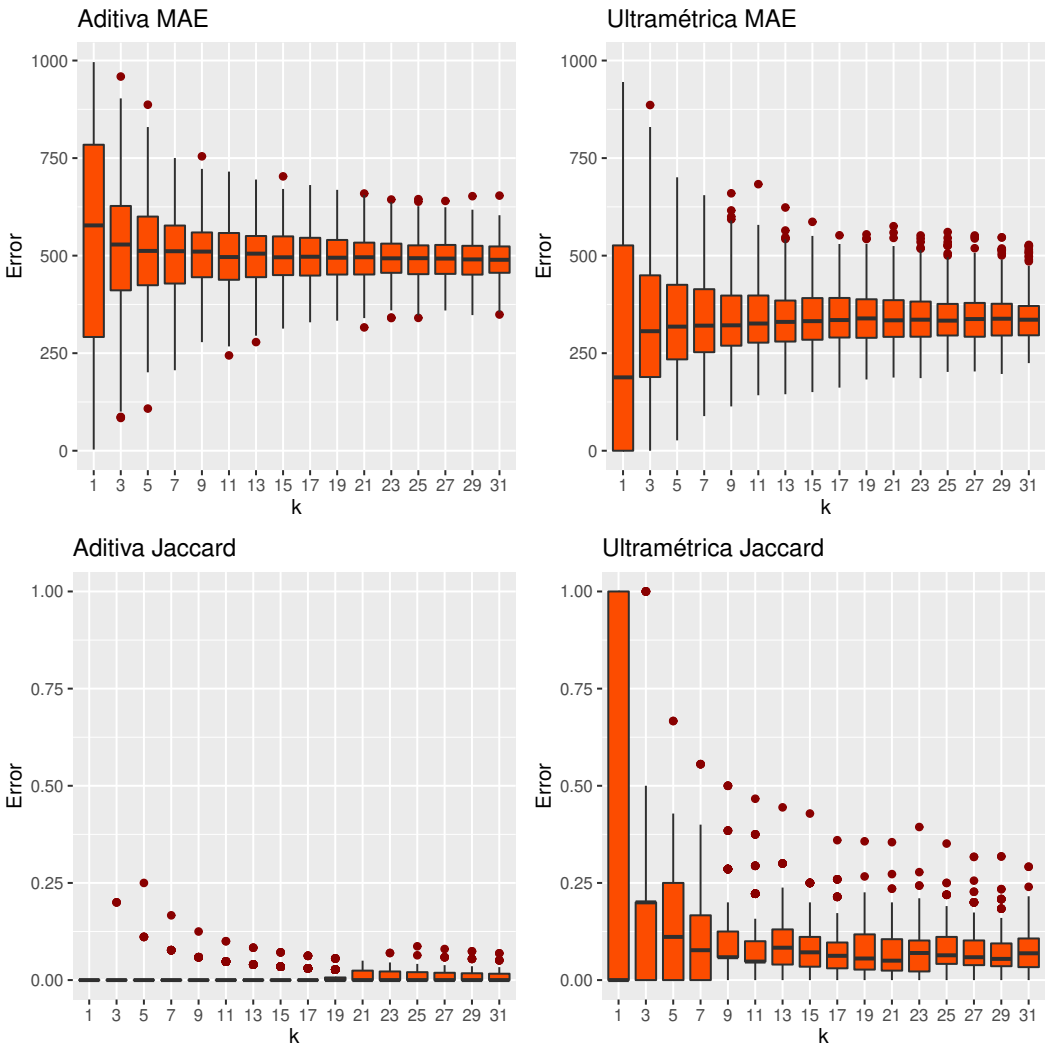

Figura 2.8: Comparación del procedimiento aditivo y ultramétrico para $n = 1000$, $l = 90\,\%$, $p = 20$ y $N = 300$.

A continuación, se muestra una comparación entre el método ultramétrico y el método propuesto basado en clústeres, donde se puede apreciar que nuestro método tiene un mejor desempeño (ver figura 2.9).

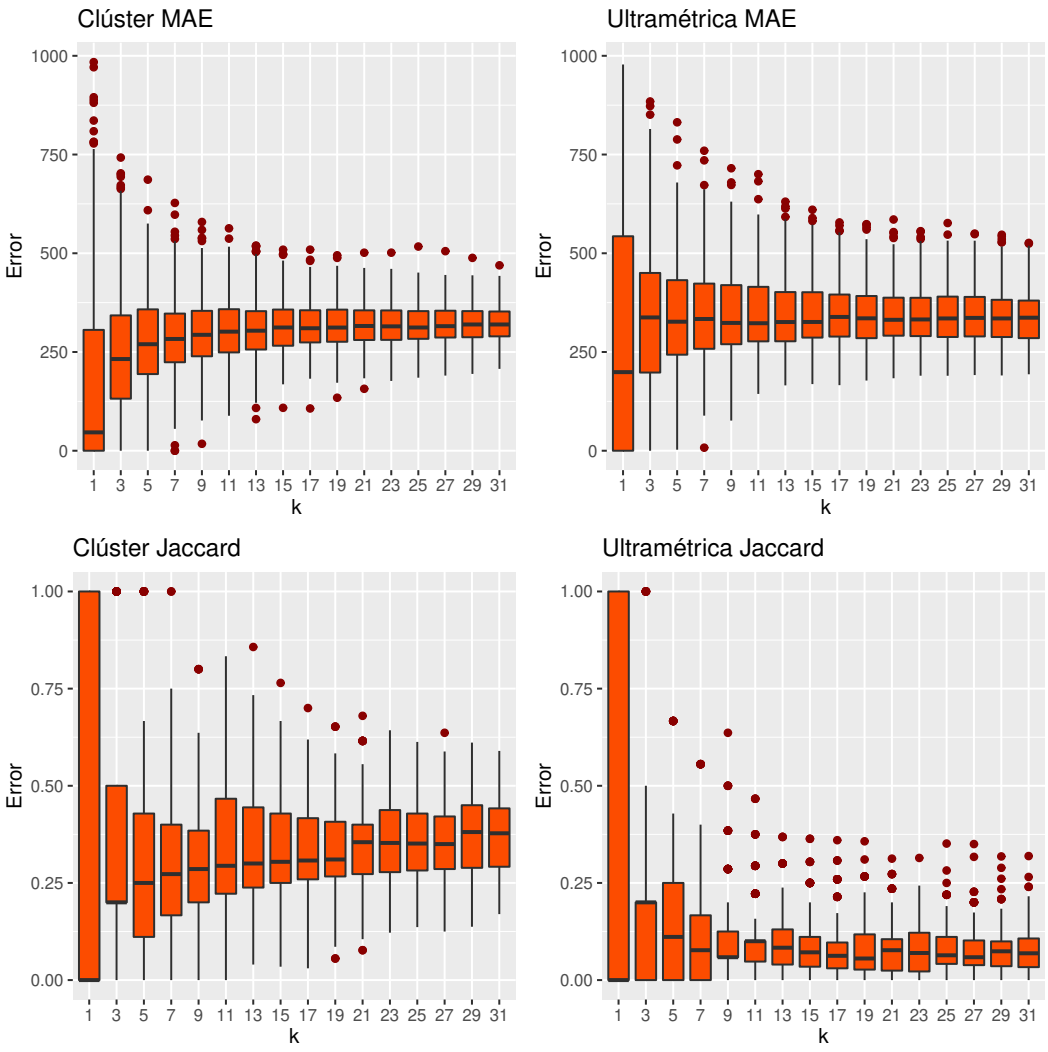

Figura 2.9: Comparación ultramétrica vs. imputación vía clúster para $n = 1000$, $l = 90\,\%$, $p = 20$ y $N = 300$.

Se observa que ambos enfoques de acuerdo al MAE no presentan una diferencia tan alta. Sin embargo, analizando el índice de Jaccard se puede ver que el método propuesto es superior al resto. Al método propuesto, en adelante, lo llamaremos como “Imputación mediante clústeres”.

3. EJEMPLOS CON CONJUNTOS DE DATOS REALES

En este capítulo, ilustraremos el uso del procedimiento de imputación mediante clústeres en dos conjuntos de datos reales muy conocidos en la literatura del aprendizaje automático:

- MNIST: Es una base de datos de dígitos escritos a mano. Son imágenes en blanco y negro.
- DIAMONDS: Es una base de datos que contiene los precios junto con otros diez atributos de varios tipos de diamantes.

3.1. MNIST

En esta sección, comprobaremos el procedimiento propuesto con el k-NN clásico usando un dataset real de clasificación. Usaremos la base de datos MNIST que consiste en imágenes en blanco y negro normalizadas cuyas dimensiones son de 28x28 píxeles en niveles de escala de grises de dígitos escritos a mano. El problema de clasificación, consiste en, dada una nueva imagen debemos predecir que número tiene escrito.

Figura 3.1: Imágenes seleccionadas del conjunto de datos de entrenamiento del MNIST.

Dicha base de datos fue descargada de Kaggle (*https://www.kaggle.com/c/digit-recognizer/data*), consta de 42000 observaciones, de las cuales dejaremos el 75 % para entrenar y validar y las restantes para probar nuestro método. Esta división se hizo usando un reparto estratificado entre las muestras de entrenamiento y de prueba [34].

Las razones para utilizar un muestreo estratificado en lugar de un muestreo aleatorio simple viene dado por [35]:

- Mantener una distribución similar de las etiquetas originales en el nuevo subconjunto.

- Para muchas aplicaciones, las mediciones se vuelven más manejables cuando la población se agrupa en estratos.
- A menudo es deseable tener estimaciones de los parámetros de la población para todos los subgrupos.

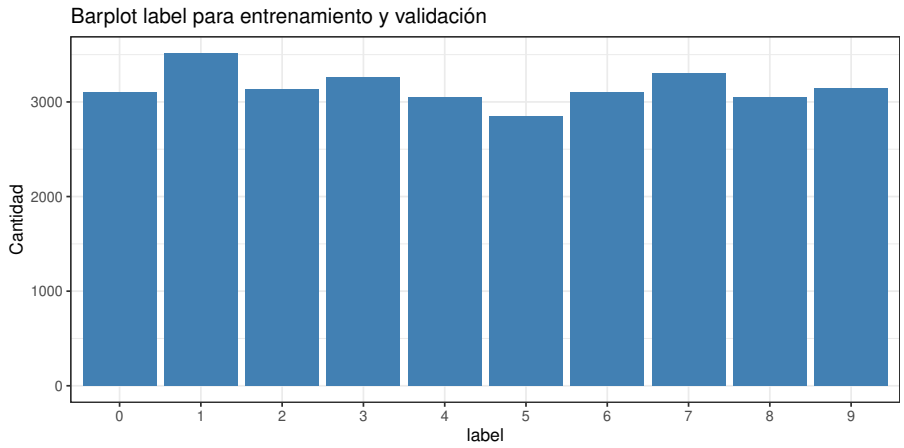

Figura 3.2: Frecuencia de las etiquetas de datos de entrenamiento y validación del MNIST.

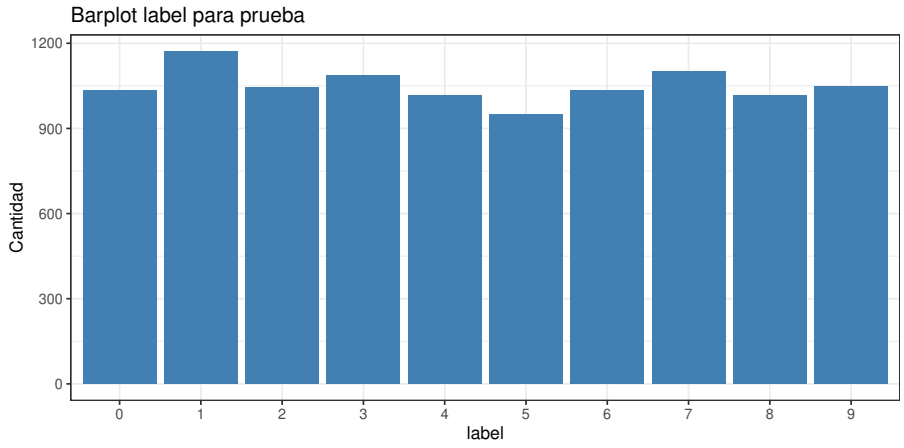

Figura 3.3: Frecuencia de las etiquetas de datos de prueba del MNIST.

Como se puede apreciar en las figuras 3.2 y 3.3 el conjunto de datos está balanceado con lo cual evitamos este problema, que es un punto débil que presenta k-NN como se había mencionado anteriormente. En caso de que fuera un dataset no balanceado se recomienda hacer data augmentation. Esta técnica es muy común a la hora de clasificar imágenes y no es más que tomar una imagen y crear nuevas muestras rotando, dilatando, trasladando o agregando ruido blanco a la original, ver figura 3.4.

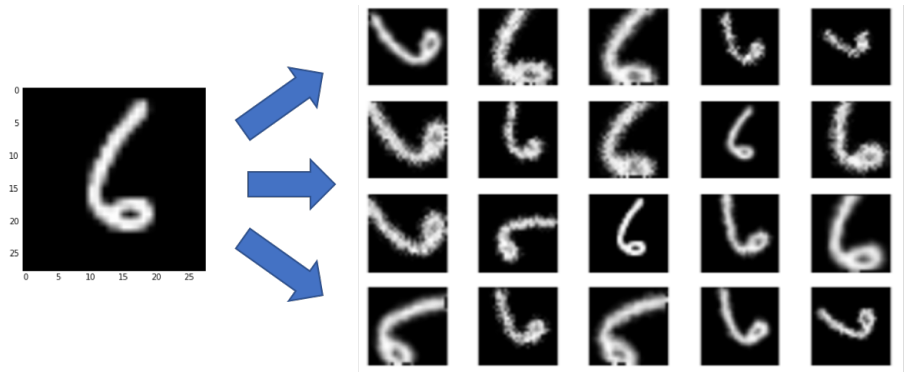

Figura 3.4: Ejemplo de data augmentation para una muestra seleccionada del MNIST.

Para la búsqueda del valor óptimo de k, se utilizó el procedimiento de validación cruzada con 5 submuestras del conjunto de entrenamiento y validación donde se obtiene una precisión del 96.14 % siendo el mejor valor $k = 3$, ver figura 3.5.

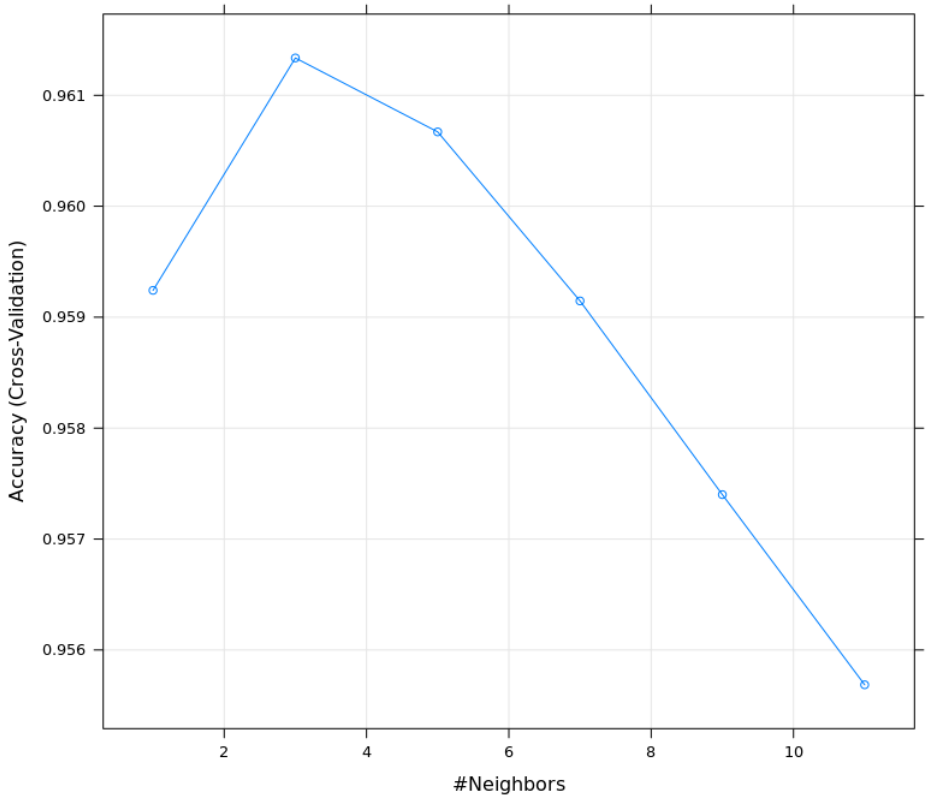

Figura 3.5: Selección del parámetro k del k-NN en datos de entrenamiento y validación del MNIST.

Usando ahora este valor de k predecimos las etiquetas para los valores de la muestra de prueba usando el k-NN donde se obtiene una precisión del 96.98 % con la siguiente matriz de confusión:

Cuadro 3.1: Matriz de confusión del MNIST usando k-NN($k = 3$).

	Referencia									
Predicción	0	1	2	3	4	5	6	7	8	9
0	1024	0	4	2	1	4	6	2	1	6
1	0	1161	8	1	7	1	0	6	10	2
2	2	4	1002	4	0	2	2	3	4	2
3	1	0	5	1043	0	8	1	0	18	10
4	0	1	0	0	985	0	0	1	3	8
5	0	1	1	20	0	920	6	0	16	6
6	5	0	0	1	8	11	1019	0	5	0
7	0	3	21	5	3	0	0	1083	5	10
8	0	1	1	6	1	0	0	0	941	1
9	1	0	2	5	13	2	0	5	12	1002

Observando la matriz anterior podemos notar que los casos donde más falla el procedimiento son en etiquetas similares, por ejemplo el 7 con el 2 y el 5 con el 8.

Para este ejercicio, trabajamos con $l = 0.75$, lo que significa que solo podemos calcular el 25 % de las distancias. Las restantes fueron imputadas usando el método propuesto, donde el número de clústeres es $K = \frac{n - ln}{2} = 3938$. De esta forma se obtiene una precisión del 96.42 % y la siguiente matriz de confusión:

Cuadro 3.2: Matriz de confusión de los valores predichos usando imputación mediante clústeres y los valores reales.

	Referencia									
Predicción	0	1	2	3	4	5	6	7	8	9
0	1022	0	9	3	2	4	11	0	4	4
1	1	1166	10	4	10	2	1	10	11	1
2	1	1	990	7	0	0	1	6	7	0
3	0	1	2	1030	0	7	0	0	14	3
4	1	2	2	0	971	0	2	6	4	6
5	0	0	0	14	0	920	2	0	19	2
6	6	0	4	3	7	8	1016	0	11	1
7	2	0	21	13	4	2	0	1067	5	12
8	0	0	3	7	0	1	1	0	923	2
9	0	1	3	6	24	4	0	11	17	1016

La matriz de confusión es similar a la obtenida utilizando todos los datos (Cuadro 3.1). Este ejercicio muestra que el procedimiento propuesto tiene un comportamiento similar al que se obtiene calculando todas las distancias.

3.2. DIAMONDS

Este conjunto de datos clásico contiene los precios y otros diez atributos de casi 54000 diamantes y está disponible en la plataforma Kaggle (*https://www.kaggle.com/shivam2503/diamonds*). El objetivo será minimizar el error absoluto medio (MAE) de la predicción del

precio en función de los atributos del diamante.

Las variables en el conjunto de datos son:

- price: Precio en dólares estadounidenses.
- carat: Peso en quilates del diamante.
- cut: Calidad del corte.
- color: Color del diamante.
- clarity: Una medición de qué tan claro es el diamante.
- x: Longitud en mm.
- y: Ancho en mm.
- z: Profundidad en mm.
- profundidad: Porcentaje de profundidad total.
- depth: Ancho de la mesa de la parte superior del diamante en relación con el punto más ancho.
- table: Ancho de la parte superior del diamante.

Cuadro 3.3: Análisis descriptivo básico de las variables numéricas de DIAMONDS.

	N	Media	Sd	Min	1Q	3Q	Max
carat	53940	0.80	0.47	0.20	0.40	1.04	5.01
depth	53940	61.75	1.43	43.00	61.00	62.50	79.00
table	53940	57.46	2.23	43	56	59	95
price	53940	3932.80	3989.44	326	950	5324.2	18823
x	53940	5.73	1.12	0.00	4.71	6.54	10.74
y	53940	5.73	1.14	0.00	4.72	6.54	58.90
z	53940	3.54	0.71	0.00	2.91	4.04	31.80

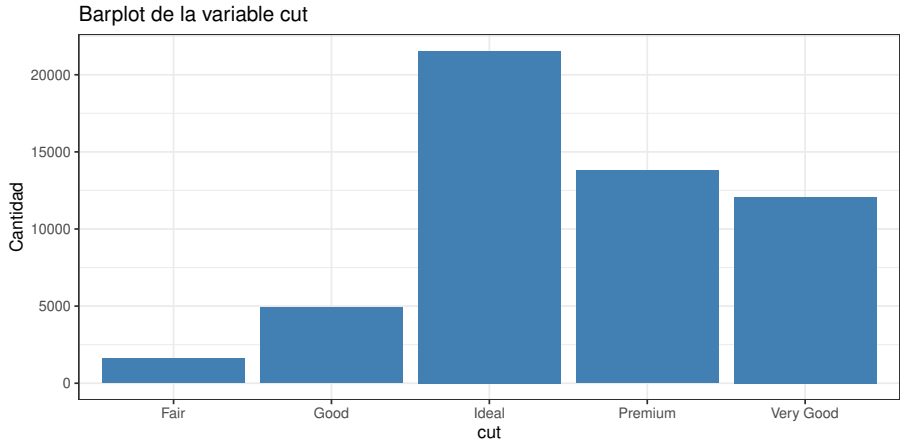

Figura 3.6: Diagrama de barras de cut.

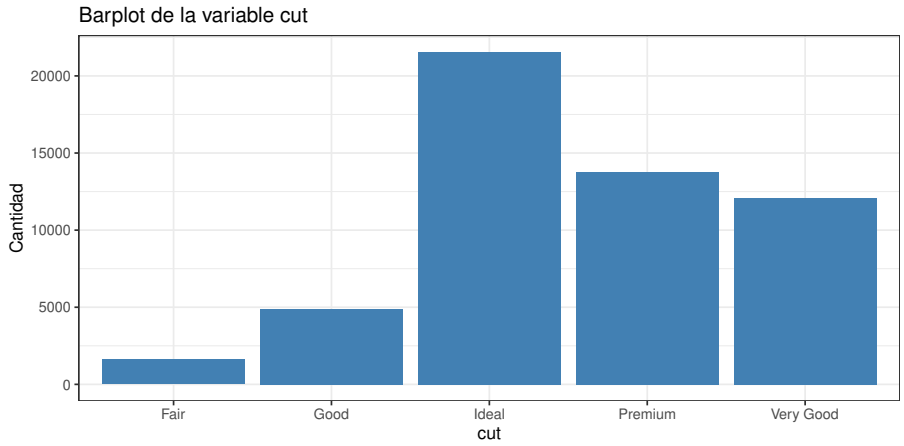

Figura 3.7: Diagrama de barras de color.

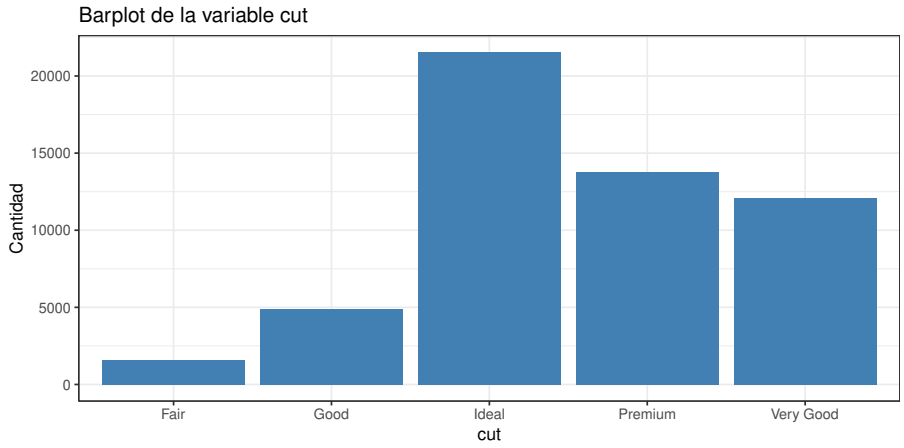

Figura 3.8: Diagrama de barras de clarity.

Igual que hicimos en el dataset anterior dejaremos el 75 %, en este caso alrededor de 40457 observaciones, para entrenar y validar y las restantes 13483 observaciones para probar nuestro método.

Cuadro 3.4: Resumen de la variable dependiente en los datos de entrenamiento y validación del dataset DIAMONDS.

Min.	1Q	Mediana	Mean	3Q	Max.
326	950	2401	3928	5324	18804

Cuadro 3.5: Resumen de la variable dependiente en los datos de prueba del dataset DIAMONDS.

Min.	1Q	Mediana	Mean	3Q	Max.
334	950	2401	3946	5324	18823

Como el objetivo es minimizar MAE, se hizo validación cruzada con 5 iteraciones para seleccionar el valor de k. En este caso se obtiene $k = 3$, ver figura 3.9.

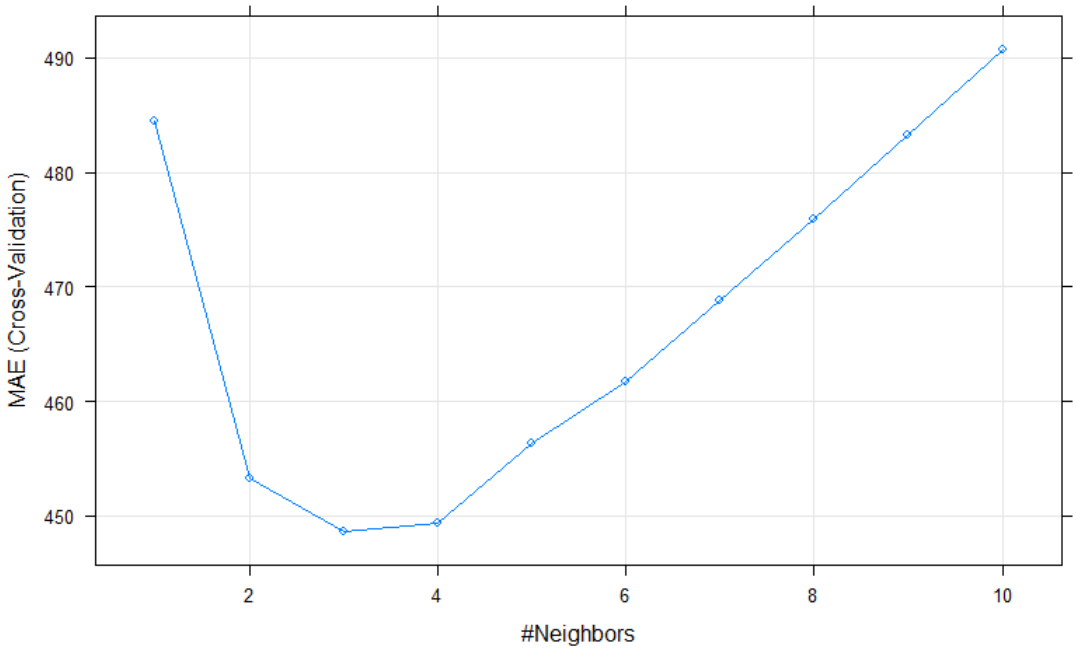

Figura 3.9: Selección del parámetro k del k-NN en datos de entrenamiento y validación de DIAMONDS.

El MAE en el conjunto de datos de prueba usando el k-NN clásico es de 432.81 y usando el procedimiento propuesto con una cantidad de clústeres de $K = \frac{n - ln}{2}$ y $l = 0.75$ se obtiene un MAE de 531.35. En este caso los resultados no son tan satisfactorios como en el MNIST. Esto puede estar dado por varias razones una de ellas puede ser que el k-NN no se comparta bien para este dataset y la otra es que no se realizó un trabajo previo de los datos antes de aplicar ambos algoritmos. A pesar de esta diferencia podemos ver en la figura 3.10 cómo se compartan ambos métodos para una selección de observaciones de los datos de prueba ilustrando que las predicciones de ambos procedimientos son coherentes.

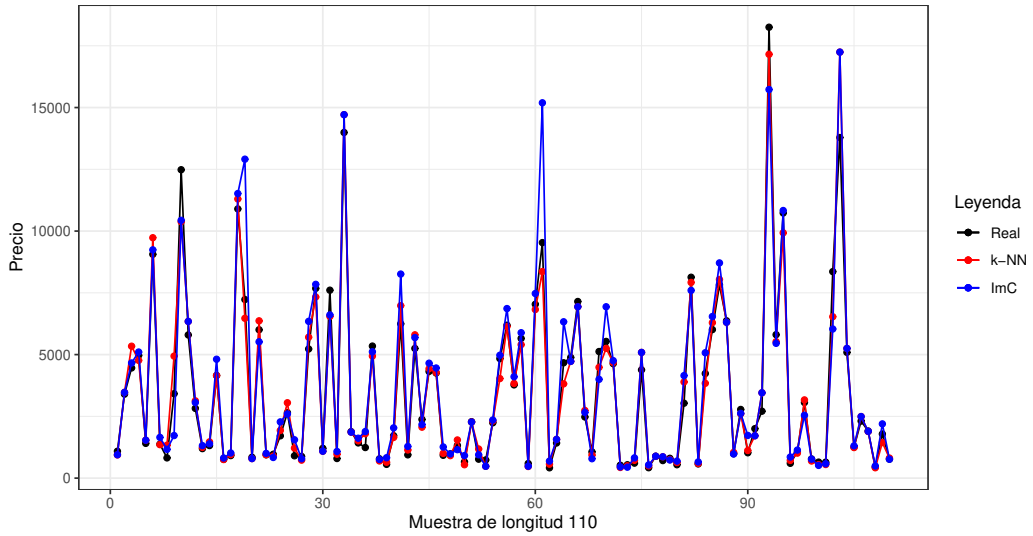

Figura 3.10: Comparación de una muestra entre k-NN, imputación mediante clústeres y valores reales de DIAMONDS

4. CONCLUSIONES Y EXTENSIONES

4.1. Conclusiones

En este trabajo fin de máster hemos estudiado modificaciones al procedimiento k-NN cuando no es factible calcular todas las distancias entre las nuevas observaciones y todas las observaciones en el conjunto de entrenamiento.

Hemos estudiado la combinación del algoritmo triangle fixing con varias propuestas de valores iniciales resultando que la propuesta basada en acotaciones obtiene mejores resultados y es computacionalmente factible en grandes conjuntos de datos.

Hemos comprobado que la selección de observaciones al azar en el conjunto de entrenamiento es inferior a una selección basada en clústeres tanto en índice de Jaccard como en el MAE.

El procedimiento de imputación basado en clústeres también resulta superior a los algoritmos aditivos y ultramétricos que han sido propuesto en relación con la inferencia filogenética.

Finalmente, hemos ilustrado la utilización del procedimiento propuesto en dos conjuntos de datos reales que muestran que en problemas de clasificación los resultados son similares al k-NN que utiliza todas las distancias y, en el caso, de regresión los resultados apuntan a que debemos refinar la selección de K para tener en cuenta la función objetivo de la regresión.

4.2. Líneas futuras de trabajo

- Implementar estos algoritmos en un lenguaje más rápido como **C** o **C++** haciendo uso del paquete *Rcpp*[36] lo que permitiría realizar experimentos en conjuntos de datos de mayor dimensión.
- Buscar un número "óptimo" de clústeres que tenga en cuenta la función de pérdida del problema a resolver con k-NN, es decir, que maximice la precisión en el caso del problema de clasificación o que minimice el error de predicción en el problema de regresión. También puede ser interesante estudiar la selección conjunta del parámetro k del k-NN y del parámetro K del procedimiento de imputación mediante clústeres.

BIBLIOGRAFÍA

[1] M. Bicego and M. Loog, "Weighted k-nearest neighbor revisited," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1642–1647.

[2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, in *When Is "Nearest Neighbor" Meaningful?* Springer Berlin Heidelberg, 1999, pp. 217–235.

[3] I. Dhillon, S. Sra, and J. Tropp, "The metric nearness problem with applications," Univ. of Texas at Austin, Tech. Rep., 2003.

[4] G. De Soete, "Additive-tree representations of incomplete dissimilarity data," *Quality and Quantity*, vol. 18, pp. 387–393, 1984.

[5] F. Lapointe and J. Kirsch, "Estimating Phylogenies from Lacunose Distance Matrices, with Special Reference to DNA Hybridization Data," *Molecular Biology and Evolution*, vol. 12, pp. 266–266, 1995.

[6] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008. [Online]. Available: http://www.R-project.org

[7] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[8] W. Rudin, *Functional Analysis*, ser. International series in pure and applied mathematics. McGraw-Hill, 1991.

[9] M. Deza and E. Deza, *Dictionary of Distances*. Elsevier Science, 2006.

[10] I. S. Dhillon, S. Sra, and J. A. Tropp, "Metric nearness: Problem formulation and algorithms," *Advances in Neural Information Processing*, vol. 17, pp. 361–368, 2005.

[11] P. Buneman, *The Recovery of Trees from Measures of Dissimilarity*. Edinburgh University Press, 1971, pp. 387–395.

[12] J. A. Hartigan, "Representation of similarity matrices by trees," *Journal of the American Statistical Association*, vol. 62, pp. 1140–1158, 1967.

[13] J. Dauben, *Georg Cantor: His Mathematics and Philosophy of the Infinite*, ser. Princeton paperbacks : History of science / Princeton paperbacks. Princeton University Press, 1990.

[14] R. Real and J. Vargas, "The Probabilistic Basis of Jaccard's Index of Similarity," *Systematic Biology - SYST BIOL*, vol. 45, pp. 380–385, 1996.

[15] J. F. Heltshe, "Jackknife estimate of the matching coefficient of similarity," *Biometrics*, vol. 44, 1988.

[16] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, pp. 679–688, 2006.

[17] J. Brickell, I. S. Dhillon, S. Sra, and J. A. Tropp, "The metric nearness problem," *SIAM J. Matrix Anal. Appl.*, vol. 30, pp. 375–396, 2008.

[18] R. T. Rockafellar, *Convex analysis*, ser. Princeton mathematical series 28. Princeton University Press, 1970.

[19] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: the dtw package," *Journal of statistical Software*, vol. 31, pp. 1–24, 2009.

[20] S. Zhang, Z. Deng, D. Cheng, M. Zong, and X. Zhu, "Efficient kNN Classification Algorithm for Big Data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[21] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, pp. 1–21, 2017.

[22] L. Rokach and O. Maimon, *Clustering Methods*. Springer US, 2005, pp. 321–352.

[23] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," pp. 4743–4759, 2018. [Online]. Available: http://cs.uef.fi/sipu/datasets/

[24] J. Han, M. Kamber, and A. K. Tung, "Spatial clustering methods in data mining," *Geographic data mining and knowledge discovery*, pp. 188–217, 2001.

[25] W. Budiaji, *kmed: Distance-Based k-Medoids*, 2019, R package version 0.2.0. [Online]. Available: https://CRAN.R-project.org/package=kmed

[26] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, pp. 3336–3341, 2009.

[27] E. Schubert and P. J. Rousseeuw, "Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms," *CoRR*, vol. abs/1810.05691, 2018.

[28] M. Mächler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik, "Cluster: Cluster analysis basics and extensions," *R packages*, vol. 1, 2012.

[29] F.-J. Lapointe and V. Makarenkov, "A weighted least-squares approach for inferring phylogenies from incomplete distance matrices," *Bioinformatics*, vol. 20, pp. 2113–2121, 2004.

[30] P.-A. Landry, F.-J. Lapointe, and J. A W Kirsch, "Estimating phylogenies from lacunose distance matrices: Additive is superior to ultrametric estimation," *Molecular Biology and Evolution*, vol. 13, pp. 818–823, 1996.

[31] P.-A. Landry and F.-J. Lapointe, *Estimation of missing distances in path-length matrices: Problems and solutions*, 1997, pp. 209–218.

[32] E. Paradis and K. Schliep, "ape 5.0: An environment for modern phylogenetics and evolutionary analyses in R," *Bioinformatics*, vol. 35, pp. 526–528, 2018.

[33] A. B. Hassanat, M. A. Abbadi, G. A. Altarawneh, and A. A. Alhasanat, "Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach," *CoRR*, vol. abs/1409.0919, 2014.

[34] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt., *caret: Classification and Regression Training*, 2019.

[35] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[36] D. Eddelbuettel and J. J. Balamuta, "Extending R with C++: A Brief Introduction to Rcpp," *The American Statistician*, vol. 72, pp. 28–36, 2018.

5. APÉNDICE: CALENDARIO DEL TRABAJO FIN DE MÁSTER

En la siguiente tabla se mostrará el tiempo invertido en la realización de este trabajo fin de máster.

Cuadro 5.1: Calendario

Objetivo	Duración	Comienzo	Fin
Estudio de la literatura	32 días	1-02-2019	5-03-2019
Análisis preliminares de métodos	40 días	6-03-2019	15-04-2019
Desarrollo y depuración de los códigos	26 días	16-04-2019	12-05-2019
Simulación	59 días	13-05-2019	11-07-2019
Análisis de datos reales	19 días	12-07-2019	31-07-2019
Escritura y revisión de la tésis	30 días	1-08-2019	31-08-2019