Universidad Zaragoza
1542

Trabajo Fin de Grado

Estrategias para agregación de información guiada por cuantificadores: aplicación a las ciudades inteligentes

"Strategies for quantifier-guided aggregation of information. An application to smart cities"

Autor/es

Álvaro Cristóbal Martínez

Director/es

Fernando Bobillo Ortega

Escuela de Ingeniería y Arquitectura

Año: 2020/2021

Resumen

La fusión del avance en las tecnologías y en la información está produciendo un cambio fundamental en el ambiente en que se desarrollan las ciudades; estas están comenzando a desarrollar planes que mejoran su competitividad, eficacia y funcionamiento, así como la calidad de vida de sus ciudadanos. Es decir, están emprendiendo el camino hacia ciudades inteligentes.

El uso de información en cuanto a datos, proporcionados por el gobierno de cada ciudad, con el fin de hacer que una ciudad sea más habitable, es un tema muy profundo, con una continua evaluación y mejoría.

En este proyecto se ha creado una propuesta de un método para la agregación de información, y una evaluación con aplicación a parámetros relevantes de las ciudades inteligentes, entendida como la agregación de diferentes evaluaciones en un único valor. Es decir, se trataría de, a partir de datos reales que muestren ciertas medidas locales, agregarlas de diferentes maneras para obtener una medida global. Es interesante hacer un estudio práctico sobre el comportamiento de todos esos mecanismos de agregación para ver cuál ofrece mejores resultados, en ciertas aplicaciones concretas.

Para ello, pese a haber muchas maneras de calcular esa agregación, se han utilizado nuevos algoritmos que permiten evaluar ciertas mediciones de ciudades. Comprobar su validez, así como evaluar las prestaciones obtenidas, será el objetivo principal de este Trabajo de Fin de Grado.

Índice

1. Introducción 4
1.1 Marco y objetivo del proyecto 4
1.2 Organización de la memoria 6

2. Contexto tecnológico 7
2.1 Operadores de agregación 7
2.1.1 Media ponderada 7
2.1.2 OWA 8
2.2 Cuantificadores 10
2.3 Montecarlo 13
2.4 Estado del arte 13

3. Diseño 16
3.1 Modelo de aprendizaje teórico 16
3.2 Búsqueda de datos 20
3.2.1 Proceso de selección con alternativas 20
3.2.2 Selección final 23
3.3 Formatos de salida 25
3.4 Toma de decisiones 28

4. Implementación y experimentación 30
4.1 Proceso de implementación 30
4.2 Evaluación de los experimentos 33
4.2.1 Consideraciones previas 33
4.2.2 Primera aplicación 34
4.2.3 Segunda aplicación 37
4.2.4 Tercera aplicación 39
4.2.5 Comparativa resultados 43

5. Conclusiones y trabajo futuro 46
5.1 Conclusiones 46
5.2 Posibles alternativas y trabajo futuro 47

Referencias bibliográficas 48

6. Anexos 52
6.1 Tiempo empleado 52
6.2 Código fuente 53

1. Introducción

1.1 Marco y objetivo del proyecto

En los últimos años se ha incrementado considerablemente el desarrollo de *Smart Cities* (ciudades inteligentes), con el objetivo de crear ciudades sostenibles económica, social y medioambientalmente. Éstas son ciudades basadas en el desarrollo urbano sostenible, que aplican la innovación y las Tecnologías de la Información y la Comunicación (TIC) a la gestión y prestación de sus diferentes servicios. Una ciudad inteligente interconecta las diferentes áreas como gobernanza, economía, movilidad, medio ambiente, energía, sanidad, seguridad, entre ellas, y permite así ser más eficiente y prestar mejores y nuevos servicios. Con ello, se pretende que las ciudades garanticen un desarrollo sostenible, un incremento de la calidad de vida de los ciudadanos, una mayor eficacia de los recursos disponibles y una participación ciudadana activa. Una armonía entre todos estos aspectos da lugar al nacimiento de la ciudad inteligente [2].

El uso de tecnologías y demás servicios está completamente integrado en la sociedad, pero no se suele fijar en ciertos detalles propios a la ciudad, los cuales la hacen más habitable. El estudio de estos factores podría suponer una mejoría en la calidad de los servicios de la ciudad, de cara a que , en base a la combinación de personas, tecnología y creatividad, se apueste por: competitividad, colaboración, planificación, creatividad y sostenibilidad [3].

La ciudad inteligente da respuesta a las necesidades cambiantes de la administración pública, las empresas y la ciudadanía a través de las nuevas tecnologías. Para la población, esto significa una mejor calidad de los servicios públicos y transparencia para una ciudad con una administración más eficiente, accesible e inclusiva. La mayor inteligencia de las ciudades evidencia también un mejor desarrollo económico, social y calidad medioambiental para las urbes y sus ciudadanos [2].

Tras el progresivo desplazamiento de la población mundial de zonas rurales a urbanas en busca de mejores trabajos y sistemas educativos, según la Organización para la Cooperación y el Desarrollo Económicos (OCDE) [4], basado en el estudio *Cities in Motion* [5], se espera que para 2050 las ciudades alberguen a casi el 70% de los habitantes del mundo, a diferencia del 50% presente en ellas en la actualidad (Naciones Unidas, 2018). Bajo este contexto nace la importancia de la tecnología, sostenibilidad,

habitabilidad y viabilidad para entender cómo van a ser las ciudades del futuro. Las ciudades necesitan obligatoriamente dar cabida a tantos millones de personas, con las mejores prestaciones y recursos posibles [6].

En la actualidad, se puede decir que las ciudades más inteligentes son Tokio, Londres, Nueva York, Zúrich, y París, según [7]. Se va a incidir en este TFG en un marco europeo, donde se va a atender a ciudades de Europa, incidiendo más en nuestro país, España, e incluso algún caso con ciudades fuera de Europa.

La agregación de diferentes criterios en un solo valor es una operación muy común en muchas aplicaciones del mundo real. Por ejemplo, las agencias de viajes en línea suelen ofrecer a los clientes una forma sencilla de comparar hoteles al combinar las puntuaciones obtenidas en diferentes criterios (como ubicación, precio o limpieza) en un solo valor. Se espera que el interés en los operadores de agregación aumente en los próximos años. Otro ejemplo sería el campo de las ciudades inteligentes, con un gran número de sensores que proporcionan piezas de información que deben combinarse de alguna manera, los operadores de agregación parecen cruciales.

Atendiendo al marco teórico, el objetivo de este TFG es la implementación en *Java* de mecanismos de agregación, de forma que, a partir de un conjunto de datos de entrada, se evalúe cada uno de ellos, analizando la viabilidad de los mismos.

El principal entorno donde se ha desarrollado esta herramienta ha sido en PC, ofrecidos por páginas oficiales como del gobierno de cada ciudad, etc. públicos al usuario, los cuales pueden consultarse y evaluarse.

Los resultados de este TFG han dado lugar a un artículo de investigación [1], el cual ha sido aceptado para su presentación en el congreso como *"Learning OWA weights by combining fuzzy quantifiers with empirical data*" (ESTYLF 20-21), y será incluido en el correspondiente libro de actas.

1.2 Organización de la memoria

La memoria está organizada en los siguientes apartados:

- **Contexto tecnológico (sección 2):** explicación detallada de conceptos y métodos utilizados a lo largo del proyecto.
- **Diseño (sección 3):** se decide cuáles son las fases requeridas para una correcta implementación y se detallan las necesidades que estas deben cumplir.
- **Implementación y experimentación (sección 4):** se desarrollan las ideas expuestas en las etapas de diseño, elaborando distintas aplicaciones y algoritmos para conseguir el objetivo final. También se evalúan las pruebas realizadas en el proceso.
- **Conclusiones y trabajo futuro (sección 5):** se comenta la validez y el alcance de los resultados, así como recomendaciones y una valoración crítica del trabajo realizado. También se hace referencia al trabajo futuro en este campo, de acuerdo a posibilidades de continuación, ampliación, mejore y/o aplicación del trabajo desarrollado.
- **Anexos (sección 6):** se plasma información adicional en relación al proyecto, a modo de ilustrar ciertos conceptos, donde a su vez se encuentran:
    - **Anexo 1**: Análisis temporal del trabajo realizado.
    - **Anexo 2**: Código fuente empleado en el proyecto.

2. Contexto tecnológico

A continuación se abordan con detalle algunos conceptos y métodos fundamentales en el desarrollo del proyecto.

2.1 Operadores de agregación

Los Operadores de Agregación (*Aggregation Operators*, AOs) son funciones matemáticas usadas para combinar diferentes contenidos de información. Los AOs calculan una salida a partir de varias entradas, típicamente con un valor de peso determinado para cada valor a agregar, de forma que obtengamos un resultado de salida, basado en la agregación de dichos parámetros de entrada, útil para el análisis posterior.

Es decir, los operadores de agregación son unos instrumentos que permiten agregar la información, es decir, a partir de una serie de datos se puede obtener un valor representativo de la información.

A continuación, se describen dos tipos de operadores, incidiendo mayormente en este Trabajo de Fin de Grado en el segundo de ellos.

2.1.1 Media y media ponderada

Se trata de un método de agregación el cual distinguimos:

- Una serie de valores $\alpha_1, \alpha_2, \dots, \alpha_n$ con típicamente $\alpha_i \in [0,1]$ (sino, se normaliza).
- Un vector de pesos determinado $W = [w_1, \dots, w_n]$ con $w_i \in [0,1]$ y $\Sigma w_i = 1$. Siendo pesos iguales (misma importancia de cada valor) para la media, y distintos para media ponderada.

A partir de los datos de entrada, el valor con el que "pese" cada uno de ellos tendrá un determinado valor, entre 0 y 1, sin atender a ningún orden predeterminado. Es decir, cada dato tendrá un peso determinado.

El resultado final de la agregación corresponde a la función $\sum_{i=1}^{n} w_i \alpha_i$

* Ejemplo 1 (media):

En un estudio de temperaturas, se obtienen cuatro sensores: 20ºC, 25ºC, 23ºC, 22ºC. Sea una ponderación de 1/n porque no se destaca una medida sobre otra:

– Agregar los valores $\alpha_1 = 20$, $\alpha_2 = 25$, $\alpha_3 = 23$, $\alpha_4 = 22$

– Vector de pesos W = [¼, ¼, ¼, ¼] = [0.25, 0.25, 0.25, 0.25]

Como resultado, queda:

$$\sum_{i=1}^{n} w_i \alpha_i = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 = 0.25*25 + 0.25*20 + 0.25*23 + 0.25*22 =$$
$$= 6.25 + 5 + 5.75 + 5.5 = 22.5$$

* Ejemplo 2 (media ponderada):

En clase, un alumno obtiene como calificaciones: Teoría: 6, prácticas: 4, trabajos: 8. Sea una ponderación del 35% en el apartado de teoría, 45% en prácticas y 20% en trabajos. Se normalizan los valores a [0,1] donde queda:

– Agregar los valores $\alpha_1 = 0.6$, $\alpha_2 = 0.4$, $\alpha_3 = 0.8$

– Vector de pesos W = [0.35, 0.45, 0.2]

Como resultado, queda:

$$\sum_{i=1}^{n} w_i \alpha_i = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 = 0.35*0.6 + 0.45*0.4 + 0.2*0.8 =$$
$$= 0.21 + 0.18 + 0.16 = 0.55$$

2.1.2 OWA

Similar al método de agregación anterior, se distinguen:

- Una serie de valores $\alpha_1, \alpha_2, \ldots, \alpha_n$ con $\alpha_i \in [0,1]$ (sino, se normaliza), siendo $\alpha_{\sigma(1)} \geq \alpha_{\sigma(2)} \geq \ldots \geq \alpha_{\sigma(n)}$ (ordenados).

- Un vector de pesos determinado $W = [w_1, \ldots, w_n]$ con $w_i \in [0,1]$ y $\Sigma w_i = 1$

En OWA (*Ordered Weighted Averaging*), se trata de funciones matemáticas empleadas para combinar información [8] [9]. A partir de los datos de entrada, el valor con el que "pese" cada uno de ellos tendrá un determinado valor, entre 0 y 1, pero esta vez sí atendiendo a un orden predeterminado.

El resultado final de la agregación corresponde a la función $\sum_{i=1}^{n} w_i \alpha_{\sigma(i)}$

Atendiendo a los mismos ejemplos de antes, los pesos W no varían, pero sí el orden de los valores α_i, ya que cada peso no se asigna a un argumento concreto, sino al que ocupe un orden determinado: es un operador no lineal. Es decir, hay un reordenamiento de los elementos, de forma que los elementos a agregar α_i no estén asociados con un peso w_1, sino que estará asociado con una posición ordenada en la agregación.

* Ejemplo 3:

– Agregar los valores $\alpha_{\sigma(1)}$, $\alpha_{\sigma(2)}$, $\alpha_{\sigma(3)}$ siguiendo una permutación:

Como $\alpha_1 = 25$, $\alpha_2 = 20$, $\alpha_3 = 23$, $\alpha_4 = 22$, se permuta a: $\sigma(1) = 1$, $\sigma(2) = 3$, $\sigma(3) = 4$, $\sigma(4) = 2$

– Vector de pesos W = [¼, ¼, ¼, ¼] = [0.25, 0.25, 0.25, 0.25]

Como resultado, queda:

$$\sum_{i=1}^{n} w_i \alpha_i = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 = 0.25*25 + 0.25*23 + 0.25*22 + 0.25*20 =$$
$$= 6.25 + 5.75 + 5.5 + 5 = 22.5$$

* Ejemplo 4:

– Agregar los valores $\alpha_{\sigma(1)}$, $\alpha_{\sigma(2)}$, $\alpha_{\sigma(3)}$ siguiendo una permutación:

Como $\alpha_1 = 0.6$, $\alpha_2 = 0.4$, $\alpha_3 = 0.8$, se permuta a: $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$.

– Vector de pesos W = [0.35, 0.45, 0.2]

Como resultado, queda:

$$\sum_{i=1}^{n} w_i \alpha_i = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 = 0.35*0.8 + 0.45*0.6 + 0.2*0.8 =$$
$$= 0.28 + 0.27 + 0.16 = 0.71$$

Escogiendo vectores de pesos apropiados, se puede modelar diferentes operadores de agregación. Si la influencia de cada valor de entrada es la misma (peso 0.25 cada uno), el resultado es el mismo que en la media clásica, pero, escogiendo diferentes vectores de pesos, el resultado será uno u otro. Todo se tratará en el siguiente apartado.

2.2 Cuantificadores

Una vez visto el método de agregación OWA anterior, se nos plantea la siguiente pregunta: "¿Cómo se pueden calcular los pesos?". Se han propuesto varias soluciones en [10], [11] y [12].

Es decir, como se ha visto en la sección 2.1.2, los métodos de agregación OWA [13] son una familia muy popular de operadores de agregación que ha sido utilizado con éxito en muchas aplicaciones [14]. Los operadores OWA están parametrizados con un vector de pesos. Mientras que la elección de dichos pesos es fundamental en el comportamiento de los operadores, la determinación de los valores concretos es un problema común en la práctica. Entre las muchas soluciones existentes, se está interesado en este proyecto en la *Quantifier Guied Aggregation* [15], donde los pesos se calculan a partir de cuantificadores difusos. Ejemplos de software o aplicaciones que utilizan dichos Q-OWA se pueden destacar Fuzzy Owl 2 [16], fuzzyDL [17], GimmeHop [18] o Fuzzy BIM [19].

El método que se va a emplear, entonces, para el aprendizaje de pesos es la agregación basada en cuantificadores. En la sección 3 se discuten otros métodos alternativos. Esta agregación basada en cuantificadores, el vector de ponderaciones W se puede definir usando un cuantificador difuso Q: [0,1]→[0,1]. Se incidirá en el caso de cuantificadores de "Crecimiento monótono regular" (RIM, "*Regular Increasing Monotone*") [20], caracterizado por la idea de que, a medida que aumenta la proporción, el grado de satisfacción no disminuye. Se puede definir este cuantificador con las siguientes propiedades [21]:

1) Conmutatividad: cualquier permutación de los argumentos tiene la misma evaluación.

2) Monotonía: Si $a_i \geq d_i$ => $F(a_1,\ldots,a_n) \geq F(d_1,\ldots,d_n)$.

3) Limitado: $\text{Min}\{a_i\} \leq F(a_1,\ldots,a_n) \leq \text{Max}\{a_i\}$.

Es decir, se dice que se cumple:

- Una monotonía creciente de los cuantificadores ($x_1 <= x_2$), lo que quiere decir que $Q(x_1) <= Q(x_2)$.

- Propiedad de internalidad. Una acotación en Q(0) = 0; Q(1) = 1.

Por ello, en los RIM, un valor 'Q' puede ser usado para definir el vector de pesos W. Con ello, cada peso, se calcula de la forma:

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right)$$ Ecuación 1. Cuantificación QOWA.

Pudiéndose observar que se cumple $w_i \in [0,1]$ y $\sum_i w_i = 1$

Dentro de estos cuantificadores, se pueden encontrar varios tipos. Para este Trabajo de Fin de Grado, se destacan tres [1]:

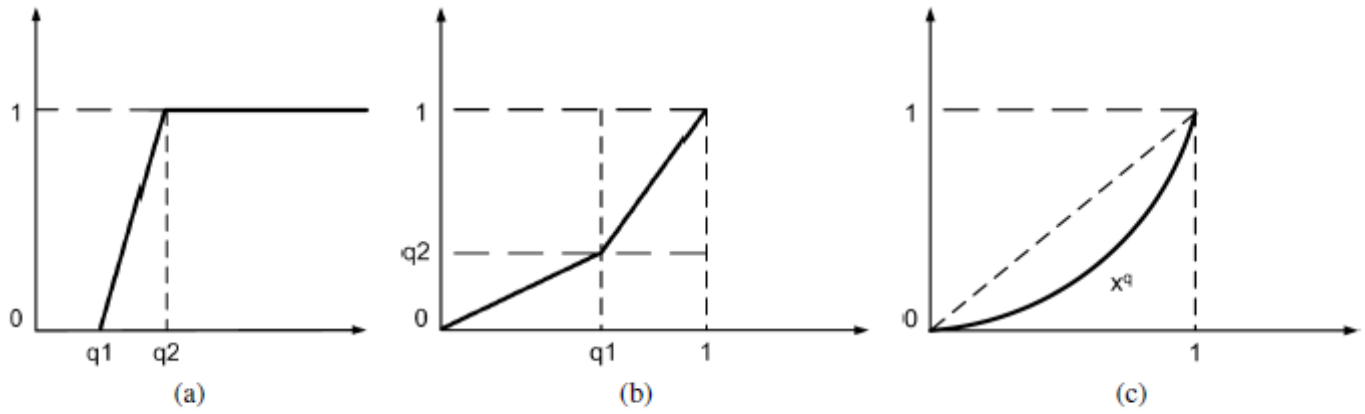


Fig. 1. (a) Right-shoulder function; (b) Linear function; (c) Power function

- Right-Shoulder (Fig 1 (a)).
  - Dados q_1; q_2 entre [0; 1] tal que $q_1 < q_2$:

$$\mathbf{right}(q_1, q_2) = \begin{cases} 0 & x \leq q_1 \\ \dfrac{x - q_1}{q_2 - q_1} & x \in [q_1, q_2] \\ 1 & x \geq q_2 \end{cases}$$

  - If $q_1 = q_2 \neq 1$:

$$\mathbf{right}(q_1, q_2) = \begin{cases} 0 & x \leq q_1 \\ 1 & x > q_1 \end{cases}$$

  - If $q_1 = q_2 = 1$:

$$\mathbf{right}(q_1, q_2) = \begin{cases} 0 & x < q_1 \\ 1 & x \geq q_1 \end{cases}$$

- Linear (Fig 1 (b)).
  - Dados q_1; q_2 entre [0; 1] con q_1 entre (0; 1):

$$\mathbf{linear}(q_1, q_2) = \begin{cases} \frac{q_2}{q_1} \cdot x & x \leq q_1 \\ \frac{(1-q_2)x + (q_2 - q_1)}{1 - q_1} & x > q_1 \end{cases}$$

  - If $q_1 = 0$:

$$\mathbf{linear}(q_1, q_2) = \begin{cases} 0 & x = 0 \\ (1-q_2)x + q_2 & x > 0 \end{cases}$$

  - If $q_1 = 1$:

$$\mathbf{linear}(q_1, q_2) = \begin{cases} q_2 \cdot x & x < 1 \\ 1 & x = 1 \end{cases}$$

- Power (Fig 1 (c)).
  - Dado q en (0;1): $\mathbf{power}(q) = x^q$

Se trata de ver cuál de estas funciones proporciona mejores resultados, y con qué parámetros 'q_i', para cada una de las aplicaciones a las que se enfrenta el TFG.

* Ejemplo 5: Cálculo del vector de pesos W con n = 5 pesos, con un cuantificador Right-Shoulder (q_1=0.1991, q_2=0.3851):

- w1 = Q(1/5) - Q(0) = 0.0048 - 0 = 0.0048
- w2 = Q(2/5) - Q(1/5) = 1 - 0.0048 = 0.9952
- w3 = Q(3/5) - Q(2/5) = 1 - 1 = 0
- w4 = Q(4/5) - Q(3/5) = 1 - 1 = 0
- w5 = Q(5/5) - Q(4/5) = 1 - 1 = 0

2.3 Montecarlo

El método de Montecarlo [22] es un método de simulación que permite calcular estadísticamente el valor final de una secuencia de sucesos no deterministas (sujetos a variabilidad), como es el caso del plazo o el coste de un proyecto.

Más técnicamente, [23] un Monte Carlo es un proceso estocástico numérico, es decir, una secuencia de estados cuya evolución viene determinada por sucesos aleatorios. Se recuerda que un suceso aleatorio es un conjunto de resultados que se producen con cierta probabilidad.

Montecarlo no explora todas las combinaciones posibles, por lo que el tiempo de ejecución es bajo (tarda menos en dar con la solución). Por contra, al no explorarlas todas, no garantiza encontrar la mejor solución. En este tipo de algoritmos se debe repetir varias veces el experimento, ya que dependen de números aleatorios.

Se presenta este método como una alternativa, a modo de comparación, de los resultados de los experimentos. La idea es generar múltiples valores aleatorios de q_1 y q_2 (o q según el caso) evaluando los experimentos, quedándonos con la pareja de valores que minimicen el error cometido (posteriormente se concretará el tipo de error).

2.4 Estado del arte

Las ciudades inteligentes juegan un papel vital en el crecimiento de la nación. La mayoría de los países ya aportan grandes inversiones en la implementación de ciudades inteligentes, lo que enfatiza su importancia. [24]

Smart City, como ya se ha comentado, es una denominación común que abarca muchos sectores, como la salud, la educación, el transporte, el agua, la energía, las comunicaciones, la seguridad y la protección, los servicios al ciudadano, etc. Todos estos sectores deben adaptarse y equiparse con tecnologías modernas como Internet de las cosas (IoT), computación en la nube y big data. La evolución de las ciudades inteligentes depende en gran medida de estas tecnologías habilitadoras clave. De ahí que sea necesario explorar los diferentes aspectos de las ciudades inteligentes en el pasado, presente y futuro.

Desde el crecimiento de las grandes poblaciones, así como el estudio de estos factores para mejorar la habitabilidad de ellas, se han tratado de investigar a lo largo de todos estos años diferentes maneras de agregación y medición.

De acuerdo a X. Liu [11], se puede enfocar el problema de búsqueda de los pesos en la agregación OWA en 5 categorías:

1. Métodos basados en la optimización matemática o programación matemática (*optimization-based methods*).

   De la definición de operador OWA, los elementos no deben ser negativos y deberían sumar la unidad. Pero esta condición no es suficiente para determinar los elementos OWA, sino que un valor orness es asignado. Sin embargo, el operador OWA con un nivel de orness dado todavía no se puede determinar de forma única a menos que sea en el caso bidimensional. Una técnica comúnmente utilizada es permitir que el operador OWA satisfaga un criterio de optimización adicional. Podrían destacarse ejemplos de métodos como el de máxima entropía y el de mínima varianza.

2. Métodos de aprendizaje de acuerdo a datos empíricos (*sample learning methods fitting to empirical data*).

   El ajuste empírico es una herramienta muy útil para la determinación del operador de agregación porque tiene una interpretación cuantitativa directa. En la mayoría de los casos, el problema de elegir el operador se traduce en algún tipo de problema de regresión, como el ajuste de mínimos cuadrados. Por lo general, se requiere un operador de agregación con ciertas propiedades para algún tipo de datos empíricos (propiedades que existen de manera natural, como por ejemplo que los pesos sumen la unidad).

3. Métodos basados en funciones (*function-based methods*).

   Método para obtener los vectores de ponderación OWA a través de cuantificadores lingüísticos difusos, o que consideran orness, especialmente el cuantificador Regular Increasing Monotone (RIM), que puede proporcionar procedimientos de agregación de información guiados por conceptos expresados verbalmente y una descripción independiente de la dimensión de la agregación deseada.

   El operador OWA, y el correspondiente cuantificador de RIM, puede verse como

el mismo operador de agregación en casos discretos y continuos respectivamente. Con la generación del cuantificador RIM, las propiedades y sus métodos de generación del operador OWA y el cuantificador RIM pueden corresponder entre sí. Si se han conocido las soluciones o las propiedades de una de ellas, también se puede anticipar la solución y las propiedades de la otra forma, y viceversa.

La familia de métodos basados en funciones incluye métodos para construir las ponderaciones a partir de un "valor orness". Por ejemplo, el vector de pesos W se puede definir partiendo de un valor deseado para el orness de dos formas recursivas, una forma recursiva izquierda y una forma recursiva derecha [25], o usando las fórmulas de Faulhaber [26]. Sin embargo, como se había comentado, el ejemplo más popular de métodos basados en funciones es la agregación basada en cuantificadores, que es el que se plasma en este TFG.

4. Métodos dependientes de argumentos (*argument dependent methods*).
   En este procedimiento, los pesos no se fijan dados valores constantes. Las ponderaciones derivadas están asociadas con posiciones ordenadas particulares de los argumentos agregados pero no tienen conexión con los argumentos agregados.

5. Métodos de preferencia (*preference methods*).
   Además de que la determinación del operador OWA considere únicamente la definición del operador OWA en sí, la determinación OWA también se puede conectar con la información de preferencia en la toma de decisiones. Estos métodos utilizan la información de preferencias como entradas, para que se puedan revelar las relaciones de preferencia entre alternativas. Con estas relaciones de preferencia, se puede construir el modelo para determinar el operador OWA más adecuado para estas relaciones de preferencia.

3. Diseño

Para llevar a cabo la implementación del algoritmo que satisfaga las necesidades del TFG, es necesaria una etapa previa de diseño. En ella se deciden cuáles son las fases requeridas para un correcto desarrollo y se detallan las necesidades que estas deben cumplir.

3.1 Modelo de aprendizaje teórico

Se van a introducir los parámetros que se van a emplear para la realización del proyecto, así como la evaluación de sus prestaciones.

Por ello, se van a definir:

- Número de datos que se evalúan: m
- Conjunto de datos de entrada (datos originales): $x_j = \langle x_{1j}, x_{2j}, \dots, x_{nj} \rangle$
- Conjunto de predicciones/estimaciones de salida OWA:

$\hat{z}_j^w = \text{OWA}(x_{1j}, x_{2j}, \dots, x_{nj}) = \langle \hat{z}_{1j}, \hat{z}_{2j}, \dots, \hat{z}_{nj} \rangle$

- Valor real de la salida: y_j

Con ello, se introduce el error MAPE (Mean Absolute Percentage Error). Mide la exactitud de un método para la construcción ajustada de valores de una serie de datos . Es decir, mide el tamaño del error (absoluto) en términos porcentuales [27] [28] [29].

$$\text{MAPE}(E, \boldsymbol{W}) = \frac{100}{m} \sum_{j=1}^{m} \left| \frac{y_j - z_j^{\boldsymbol{W}}}{y_j} \right|$$ Ecuación 2. Error MAPE

Se puede observar en la Ecuación 2 que es el promedio del error absoluto o diferencia entre la demanda real y el pronóstico, expresado como un porcentaje de los valores reales [30]. Con ello, nótese que las series de estimaciones y la salida real tienen que tener el mismo tamaño.

El mejor de los casos es cuando MAPE = 0 (error nulo). MAPE altos quieren decir gran desviación entre lo predicho y lo real. El peor caso, es cuando MAPE=100. Se quedaría con el cuantificador, y vector W que minimicen el MAPE (Ecuación 3).

$$\underset{\boldsymbol{W}}{\text{argmin}}\ \text{MAPE}$$ Ecuación 3. MAPE(E,W) mínimo

Por ello, de cada uno de los métodos de agregación (Right-Shoulder, Linear y Power) se obtiene un valor de MAPE para cada uno de los posibles valores de q_1 y q_2.para Se acumula el error, de tal forma que se tendrá que elegir cuál de los tres cuantificadores, y con qué parámetros 'q_i', ofrece mayor exactitud (menor error). Véase el Ejemplo 6.

* Ejemplo 6:

Se obtiene para febrero, en un cuantificador Right-Shoulder con $q_1 = 0.5539$ y $q_2 = 0.6783$, una predicción a partir de 5 datos de entrada (5 ciudades), una temperatura estimada de una sexta ciudad de 6.70384ºC. Se sabe que el valor real en febrero en esa ciudad es de 6'8ºC.

Por tanto, el error MAPE vendrá dado por: |(6.8-6.70384)/6.70384| = 0.0143.

Ese error se iría acumulando, para cada combinación de q_1 y q_2, dando un valor MAPE total para el método Right-Shoulder utilizado. Se trata de ver cuál de los métodos empleados es el más adecuado para cada aplicación. Es decir, seleccionar la opción que minimice el error.

La clave de este enfoque es que W se construye utilizando diferentes tipos de funciones RIM y con diferentes parámetros, siempre y cuando el espacio de búsqueda no sea muy grande, y calcular los pesos y la agregación usando OWA no son operaciones computacionalmente costosas. Por tanto, podría ser posible calcular los mejores parámetros por fuerza bruta, al menos para conjuntos de datos no muy grandes.

En el Algoritmo 1, se muestra cómo calcular los parámetros de un cuantificador RIM Right-Shoulder que minimiza el MAPE usando fuerza bruta. El algoritmo recorre los valores de q_1 y q_2 agregando un incremento Δ. El mejor MAPE se inicializa para 100, el valor más alto posible. Cuando una se encuentra un MAPE más pequeño, los parámetros q_1 y q_2 que hecho posible se almacenan. Finalmente se devuelven $\langle q_1, q_2 \rangle$.

En el Algoritmo 2, de manera similar pero para un cuantificador RIM Linear, parecido pero el bucle sobre todos los valores de q_2 no comienza desde q_1 ($q_1 \leq q_2$).

En el Algoritmo 3, para un cuantificador RIM Power, se necesita un solo bucle, ya que solo hay un parámetro q, que va en $(0, \infty)$. Requiere que > 0.

Algorithm 1 Brute-force algorithm to compute the parameters of a RIGHT-SHOULDER quantifier minimizing the MAPE.

Input: A dataset E with examples
Output: Parameters $\langle q_1, q_2 \rangle$ of a right-shoulder function.

```
1: bestMapeRS ← 100
2: for q1 ← 0 to 1 by Δ do
3:   for q2 ← 0 to 1 by Δ do
4:     if q1 < q2 then
5:       Q ← rightshoulder(q1, q2)
6:       W ← compute a vector from Q
7:       mape ← compute MAPE(E, W)
8:       if mape < bestMapeRS then
9:         bestMapeRS ← mape
10:        bestQ1 ← q1
11:        bestQ2 ← q2
12:      end if
13:    end if
14:  end for
15: end for
16: return ⟨bestQ1, bestQ2⟩
```

Algorithm 2 Brute-force algorithm to compute the parameters of a LINEAR quantifier minimizing the MAPE.

Input: A dataset E with examples
Output: Parameters $\langle q_1, q_2 \rangle$ of a linear function.

```
1: bestMapeL ← 100
2: for q1 ← 0 to 1 by Δ do
3:   for q2 ← 0 to 1 by Δ do
4:     Q ← linear(q1, q2)
5:     W ← compute a vector from Q
6:     mape ← compute MAPE(E, W)
7:     if mape < bestMapeL then
8:       bestMapeL ← mape
9:       bestQ1 ← q1
10:      bestQ2 ← q2
11:    end if
12:  end for
13: end for
14: return ⟨bestQ1, bestQ2⟩
```

Algorithm 3 Brute-force algorithm to compute the parameters of a POWER quantifier minimizing the MAPE.

Input: A dataset E with examples
Output: Parameters $\langle q_1, q_2 \rangle$ of a power function.

```
1: bestMapeP ← 100
2: for q ← 0 to 1 by Δ do
3:   Q ← power(q1, q2)
4:   W ← compute a vector from Q
5:   mape ← compute MAPE(E, W)
6:   if mape < bestMapeP then
7:     bestMapeP ← mape
8:     bestQ ← q
9:   end if
10: end for
11: return ⟨bestQ⟩
```

Nótese que:

- No se dividen los datos en conjuntos de entrenamiento, validación y test.

- Se separa en tres el problema, pues para los resultados deseados se va a recurrir a cada método según los valores de q_1 y q_2:

* El método Power se realiza siempre que q_1 ($q=q_1$) esté entre 0 e “Infinito”.

* El método Linear se realiza siempre que q_1 y q_2 estén entre 0 y 1.

* El método Right-Shoulder se realiza siempre que q_1 y q_2 estén entre 0 y 1 además de que q_1 sea menor que q2. En caso contrario, dicha función no está definida.

Puede corresponderse este esquema a lo siguiente:

```
Escribir q1 y q2
if (q1 < q2)
{
    RIGHT-SHOULDER(q1,q2) (escritura resultado)
}
LINEAR(q1, q2) (escritura resultado)
if (q2 == 0.0)
{
    POWER(q1) (escritura resultado)
}
```

Para grandes conjuntos de datos, podría ser posible utilizar métodos heurísticos, como algoritmos Monte Carlo, búsqueda local o algoritmos evolutivos. Por eso, se emplea también, a modo de comparativa, el Monte Carlo, representado en el Algoritmo 4 (por ejemplo para método Linear):

Algorithm 4 Monte Carlo algorithm to compute the parameters of a linear quantifier minimizing the MAPE.

Input: A dataset E with examples and the number of repetitions MAX_REPETITIONS.
Output: Parameters $\langle q_1, q_2 \rangle$ of a linear function.

```
bestMapeL ← 100
repetition ← 0
repeat
    q1 ← random number in [0, 1]
    q2 ← random number in [0, 1]
    Q ← linear(q1, q2)
    W ← compute a vector from Q
    mape ← compute MAPE(E, W)
    if mape < bestMapeL then
        BestMapeL ← mape
        BestQ1 ← q1
        BestQ2 ← q2
    end if
    repetition ← repetition + 1
until repetition = MAX_REPETITIONS
return ⟨bestQ1, bestQ2⟩
```

La idea es generar números pseudo-aleatorios con los posibles valores de q_1 y q_2, repitiendo el experimento varias veces, y guardar los valores que minimizan el MAPE.

Para concluir este apartado, se observa que es trivial considerar el caso en el que faltan valores, es decir, para algunos ejemplos, algunos de los valores x_{ji} son desconocidos. En este caso, en lugar de tener el mismo vector W para todos los ejemplos, se podría calcular para cada ejemplo x_j un vector de ponderación de dimensión n_j , dónde norte n_j es el número de valores no vacíos para las variables de entrada, usando el mismo tipo de función y parámetros para todos los ejemplos.

3.2 Búsqueda de datos

En este apartado, se pretende plasmar el proceso de encontrar datos con los que realizar el proyecto. ¿Por qué? Porque para evaluar cómo de buena es cada una de las medidas globales que se calculen, haría falta también tener como parte del conjunto de datos un valor de referencia objetivo. Por lo tanto, para que esto sea viable se necesita un conjunto de datos adecuado.

Se trata de buscar datos individuales (de cada sensor) de alguna medida tal que la información de diferentes sensores se pueda agregar. Además, se necesitan datos del valor global ya agregado, para ver si coinciden con las diferentes estrategias de agregación que se propongan.

Por tanto, a modo de poner en contexto, la idea es buscar un dataset donde haya valores individuales y valores agregados. Con eso, se pueden ver diferentes maneras de fusionar los datos individuales en datos agregados y estimar el porcentaje de acierto de cada una, para ver cuál sería la mejor manera de fusionar. Ya se conocen diferentes técnicas de fusión, sería "solo" aplicarlas. Pero claro, para hacer eso, es necesario disponer de esos datos agregados, que es de lo que se va a tratar en este apartado.

3.2.1 Proceso de selección

Lo primero es pensar si tiene sentido agregar ciertos datos. Es decir, para cada una de las aplicaciones, tiene que tener cabida en el ámbito de las Smart Cities.

También puede haber casos donde no se quiere agregar la misma magnitud (temperatura) de todos los sensores, sino una magnitud de uno, otra de otro, etc. Por ejemplo, saliendo del ámbito de los sensores, sería a partir de la puntuación en limpieza, amabilidad del personal, ubicación, servicios, etc. calcular una puntuación global para un hotel.

Como primera selección, se ha seleccionado, vistos en la Figura 1:

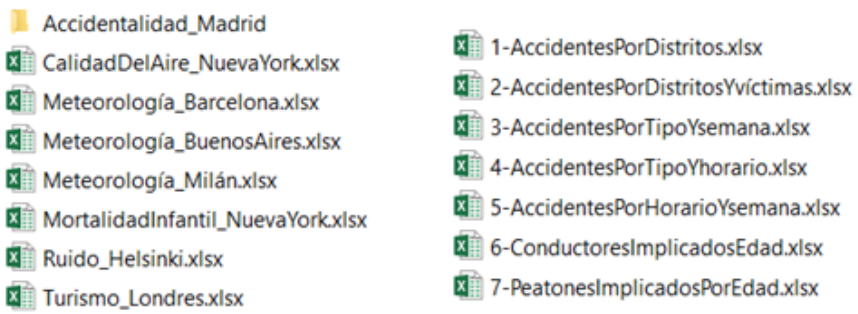


Figura 1. Primera selección de datos

- Accidentalidad (Madrid) [31] (Figura 2): Mide la accidentalidad según distintos parámetros (el distrito, las víctimas, el tipo de accidente, por edad del implicado…)

- Calidad del aire (Nueva York) [32] (Figura 3): Clasificación de distintos parámetros (nivel de CO2, nivel de partículas…) según la zona de la ciudad.

- Meteorología (Barcelona, Buenos Aires, Milán): Datos en relación a temperaturas, precipitaciones, niveles de viento, humedad...

- Mortalidad infantil (Nueva York) [33] (Figura 4): Recuento del número de muertes en base a la raza y año de suceso.

- Ruido (Helsinki) [34] (Figura 5): Niveles de ruido generados en base a distintos sensores distribuidos a lo largo de la ciudad.

- Turismo (Londres): Datos en relación al turismo de la ciudad (número de alojamientos, total gastado…)

Indicadores	Nº Accidentes											
Año	2016											Total
DISTRITO_ACCIDENTE	COLISIÓN DOBLE	COLISIÓN MÚLTIPLE	CHOQUE CON OBJETO FIJO	ATROPELLO	VUELCO	CAÍDA MOTOCICLETA	CAÍDA CICLOMOTOR	CAÍDA BICICLETA	CAÍDA VIAJERO BUS	OTRAS CAUSAS	CAÍDA VEHÍCULO 3 RUEDAS	
ARGANZUELA	328	62	86	57	1	46	12	10	6	2		610
BARAJAS	95	6	50	34	4	12	2	4	2	4		213
CARABANCHEL	346	55	121	106	5	53	18	7	8	6		725
CENTRO	444	23	172	144		97	11	33	5	3		932
CHAMARTIN	499	59	122	80	5	105	14	8	5	7		904
CHAMBERI	382	26	53	88		63	10	19	1	5		647
CIUDAD LINEAL	411	57	119	85	4	75	15	8	5	8		787
FUENCARRAL-EL PARDO	345	30	132	70	6	56	18	28	3	4		692
HORTALEZA	248	19	90	56	5	33	9	10	1	1		472
LATINA	288	40	119	104	3	36	12	11	7	2		622
MONCLOA-ARAVACA	294	61	182	54	12	68	8	27	2	1		709
MORATALAZ	150	31	53	32	5	23	5	2	4	2		307
PUENTE DE VALLECAS	386	81	140	118	8	41	11	11	10	5		811
RETIRO	344	69	63	56	1	67	9	33	1	3	1	647
SALAMANCA	467	65	111	112	1	142	14	22	14	3		951
SAN BLAS	299	20	57	68	3	34	6	9	5			501
TETUAN	330	43	70	91		73	25	13	5	3		653
USERA	206	27	74	75	2	16	2	5	4	1		412
VICALVARO	70	4	38	22	4	15	2	3	2	1		161
VILLA DE VALLECAS	149	7	42	54	5	26	4	4	1	3		295
VILLAVERDE	177	11	71	54	1	8	5	6	2	1		336
Total	6.258	796	1.965	1.560	75	1.089	212	273	93	65	1	12.387

Figura 2. Accidentalidad Madrid

	A	B	C	D	E	F	G	H	I	J	K	L
1	unique_id	indicator_i	name	measure	measure_i	geo_type_	geo_join_id	geo_place	time_perio	start_date	data_value	message
2	333939	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Winter 20	01/12/2014	10,21	
3	547354	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Annual Ave	01/01/2017	7,72	
4	605650	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Winter 20	01/12/2017	8,28	
5	179503	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Winter 20	01/12/2010	13,85	
6	179643	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Annual Ave	01/12/2008	11,05	
7	179648	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Annual Ave	01/12/2009	10,14	
8	179568	365	Fine Partic	Mean	mcg per cu	Borough	1	Bronx	Summer 2(	01/06/2009	10,73	

Figura 3. Calidad del aire Nueva York

	A	B	C	D	E	F	G	H	I	J
1	Year	Materal Race or Ethnicity	Infant Mortality Rate	Neonatal Mortality Rate	Postneonatal Mortality Rate	Infant Deaths	Neonatal Infant Deaths	Postneonatal Infant Deaths	Number of Live Births	
2	2007	Black Non-Hispanic	9,8	6	3,8	287	177	110	29,268	
3	2013	Other Hispanic	4,3	2,6	1,7	120	72	48	27,621	
4	2013	Black Non-Hispanic	8,3	5,5	2,9	201	132	69	24,108	
5	2008	White Non-Hispanic	3,3	2,1	1,1	125	82	43	38,383	
6	2009	Black Non-Hispanic	9,5	5,8	3,7	259	158	101	27,405	
7	2010	Black Non-Hispanic	8,6	5,6	3,1	230	148	82	26,635	
8	2010	White Non-Hispanic	2,8	2	0,8	104	75	29	37,78	
9	2011	Black Non-Hispanic	8,1	5,3	2,9	210	136	74	25,825	

Figura 4. Mortalidad infantil Nueva York

	A	B	C	D
1	time	readable_time	dBA	dev-id
2	1,57775E+12	2019-12-31T00:00:10Z	53,6	TA120-T246191
3	1,57775E+12	2019-12-31T00:00:20Z	49,7	TA120-T246187
4	1,57775E+12	2019-12-31T00:00:44Z	59,8	TA120-T246189
5	1,57775E+12	2019-12-31T00:00:57Z	64	TA120-T246184
6	1,57775E+12	2019-12-31T00:01:10Z	54,1	TA120-T246191
7	1,57775E+12	2019-12-31T00:01:20Z	48,7	TA120-T246187
8	1,57775E+12	2019-12-31T00:01:44Z	60,6	TA120-T246189
9	1,57775E+12	2019-12-31T00:01:57Z	63,4	TA120-T246184
10	1,57775E+12	2019-12-31T00:02:10Z	50,5	TA120-T246191
11	1,57775E+12	2019-12-31T00:02:20Z	49,1	TA120-T246187
12	1,57775E+12	2019-12-31T00:02:44Z	59,5	TA120-T246189
13	1,57775E+12	2019-12-31T00:02:57Z	60,5	TA120-T246184
14	1,57775E+12	2019-12-31T00:03:10Z	55,3	TA120-T246191
15	1,57775E+12	2019-12-31T00:03:20Z	51,3	TA120-T246187

Figura 5. Nivel de ruido Helsinki

Un problema encontrado es que, para algunos datasets, los datos no se pueden descargar directamente a un fichero, únicamente se pueden consultar.

Otra cuestión es que estaría bien que los datasets, aunque se puedan descargar para trabajar localmente, puedan consultarse vía web. Típicamente, se ofrecen puntos de acceso para hacer preguntas y obtener datos en directo, que se actualicen de manera dinámica. Así, sin necesidad de descargar datos, realmente se estaría interactuando con smart cities.

Por tanto, ¿se ha pensado algún ejemplo de variables para agregar y resultados de agregación? Recordando, la idea es:
Teniendo una variables de entrada e1, e2, ... , eN
Con una función f(e1, e2, ..., eN) se calcula el valor de la variable de salida.

* Ejemplo 7:

A partir de la temperatura de Milano-Brera (e1) y la de Milano-Lambrate (e2), se calcula la de la plaza de la catedral Milano-Duomo (s) con un método OWA: s = e1 * 0.67 + e2 * 0.33

En base a esto, se analizan las opciones:

- Accidentalidad (Madrid): Se tiene una serie de clasificaciones, por lo que es adecuado, pero se necesitan mecanismos más sofisticados (redes neuronales, random forests, …) ya que hay que fijar el valor de los pesos, no siendo el objetivo de este TFG.
- Calidad del aire (Nueva York): Podría ser una aplicación. Pero es muy analítico. Mide de forma numérica sólo una clasificación (el nivel). El resto, son entradas que podrían dar lugar a pequeños datasets dentro del dataset mayor. Pero no puede usarse como única clasificación.
- Meteorología (Barcelona, Buenos Aires, Milán): Se puede elegir. Pero hay que elegir bien, pues se necesitan datos de entrada y de salida de la misma forma. Es decir, que los datos se clasifiquen por día, por año… que tengan las mismas unidades, etc.
- Mortalidad infantil (Nueva York): No se puede tener como variable de entrada únicamente el año, y como variable de salida cualquiera del resto.
- Ruido (Helsinki): Se tienen muchos datos, pero solo con una clasificación (nivel de ruido). Si se tuvieran más, se podría clasificar. Pero mostrando el nivel de ruido por hora no se puede.
- Turismo (Londres): Puede ser escogido como aplicación, puesto que es un tema muy influyente en lo que a la smart city se refiere, y se tiene solamente tres clasificaciones para muchas entradas (muchas filas de datos). En este caso, el peso correspondiente a cada valor se va a distinguir mejor. Por contra, si se tuvieran muchas clasificaciones (columnas) pero pocas entradas, los pesos serían muy pequeños y sería difícil ver qué valor obtiene mayor o menor peso respecto al resto en el resultado.

3.2.2 Selección final

Teniendo todo lo anterior en cuenta, finalmente la selección de los conjuntos de datos a evaluar van a ser los siguientes:

- Dataset número 1 (Figura 6): Temperaturas medias globales de ciudades en el año de 2018. En concreto de Buenos Aires [35], Milán [36], Barcelona [37], Madrid [38], Basilea [39], Logroño [40] y Zaragoza [41].
  Número de variables de entrada (columnas): 6.
  Número de ejemplos (filas): 12.

	BUENOS AIRES	MILÁN	BARCELONA	MADRID	BASILEA	LOGROÑO	ZARAGOZA
ENERO	26,1	6	10,5	4,8	6,8	7,6	9
FEBRERO	25,35	7,5	6,7	6	1,4	5,5	6,8
MARZO	22,5	11,9	10,8	9,2	6,9	8,9	10,7
ABRIL	22,15	15,4	14,7	12,2	14,9	12,7	14,9
MAYO	16,75	18,4	17,1	16,7	17,3	15,1	18,2
JUNIO	11,3	22,9	21,5	22,8	20,2	19,7	22,7
JULIO	10,75	22,6	25,3	26	22,9	22,7	27
AGOSTO	12,5	22,4	25,8	25,6	22,7	22,9	26,7
SEPTIEMBRE	17,65	20,2	22,5	20,9	18,8	20,7	23,6
OCTUBRE	17,8	16,4	17	15,1	13,6	141	17,6
NOVIEMBRE	21,5	11,3	12,4	8,6	8,4	9,8	12,1
DICIEMBRE	22,5	6,3	11,1	5,5	6,1	8,6	9,26

Figura 6. Dataset empleado para los experimentos 1 y 2

- Dataset número 2 (Figura 7): Temperaturas de ciudades españolas (Madrid, Barcelona, Sevilla, Valencia, Cádiz, Oviedo y Zaragoza), desglosadas desde 2014-2018 [42].

  Número de variables de entrada (columnas): 6.

  Número de ejemplos (filas): 60.

		MADRID	BARCELONA	SEVILLA	VALENCIA	CÁDIZ	OVIEDO	ZARAGOZA
2014	ENERO	8,08	11,04	12,29	14,47	13,7	9,46	9,16
	FEBRERO	7,91	11,33	12,35	12,91	13,23	9,36	9,54
	MARZO	10,99	12,98	15	13,79	14,82	10,17	12,28
	ABRIL	15,99	16,18	19,02	18,57	17,84	13,12	17,21
	MAYO	18,2	17,67	22,38	19,52	20,62	13,65	18,35
	JUNIO	22,36	22,2	24,64	23,74	22,08	17,75	23,59
	JULIO	24,25	23,84	26,2	25,55	23,09	18,98	24,1
	AGOSTO	25,51	24,81	27,64	26,53	23,9	18,79	24,98
	SEPTIEMBRE	20,99	23,37	24,66	24,71	23,2	19,65	23,45
	OCTUBRE	18,26	20,36	22,75	21,69	22,12	17,7	19,91
	NOVIEMBRE	11,17	16,4	16,38	15,88	17,29	11,77	12,74
	DICIEMBRE	6,55	10,44	10,98	10,74	12,48	8,39	8,45
2015	ENERO	5,53	10,11	10,63	10,63	12,01	7,7	7,41
	FEBRERO	6,65	9,47	11,29	10,12	12,23	6,83	7,31
	MARZO	10,93	13,117	16,1	14,53	14,85	10,12	12,65
	ABRIL	14,31	15,25	18,8	14,55	17,51	13,56	16,05
	MAYO	19,49	19,13	24,04	10,14	21,93	14,92	20,24
	JUNIO	23,92	23,48	26,32	23,08	23,45	17,87	24,61
	JULIO	28,94	27,12	30,27	27,52	25,87	20,02	27,82
	AGOSTO	25,5	25,33	28,32	26,48	24,46	19,5	25,7
	SEPTIEMBRE	20,43	21,53	23,94	22,1	22,25	16,55	20,69
	OCTUBRE	15,75	18,11	20,66	18,68	20,47	14,98	16,33
	NOVIEMBRE	11,73	14,61	16,22	14,55	17,2	13,18	12,72
	DICIEMBRE	8,43	12,55	14,27	12,12	16,3	12,89	7,99
2016	ENERO	8,24	12,4	13,3	12,43	14,87	10,54	10,13

Figura 7. Dataset empleado para el experimento 3

- Dataset número 3 (Figura 8): Datos en relación al turismo de Londres entre 2002 y 2019 [43].
  Número de variables de entrada (columnas): 2.
  Número de ejemplos (filas): 72.

Year	Quarter	Total Visits (000s)	Total Nights (000s)	Total Spend (£m)	Sample size	Total Visits	Total Nights	Total Spend
2002	Q1	2.195	13.864	£1.084	4.900	0,3413	0,3495	0,2303
2002	Q2	2.996	17.793	£1.443	5.905	0,466	0,4486	0,3065
2002	Q3	3.502	25.009	£1.749	6.270	0,5446	0,6306	0,3715
2002	Q4	2.911	18.736	£1.512	5.284	0,4528	0,4724	0,3213
2003	Q1	2.530	15.858	£1.179	4.826	0,3935	0,3998	0,2504
2003	Q2	2.796	17.227	£1.342	5.221	0,4348	0,4344	0,285
2003	Q3	3.446	25.908	£1.859	5.628	0,536	0,6533	0,3948
2003	Q4	2.925	19.954	£1.491	5.231	0,4549	0,5031	0,3167
2004	Q1	2.736	15.645	£1.286	5.206	0,4256	0,3945	0,2731
2004	Q2	3.371	22.619	£1.646	6.028	0,5243	0,5703	0,3496
2004	Q3	3.868	30.362	£1.834	6.204	0,6017	0,7656	0,3896
2004	Q4	3.414	21.611	£1.674	5.871	0,531	0,5449	0,3556
2005	Q1	3.091	19.187	£1.446	5.595	0,4808	0,4838	0,3071
2005	Q2	3.704	20.958	£1.707	6.465	0,5762	0,5284	0,6327
2005	Q3	3.723	28.852	£1.844	6.187	0,5792	0,7275	0,3918
2005	Q4	3.374	22.846	£1.861	5.747	0,5248	0,5761	0,3954

Figura 8. Dataset empleado para los experimentos 4, 5 y 6

3.3 Formatos de salida

La salida de los resultados, que en la sección 4 se desarrollarán, se van a plasmar en tres estilos:

- Desglosado (Figuras 9, 10): Se genera un archivo (.txt y/o .xlsx) por cada pareja de valores q_1 y q_2, de forma que se encuentre en dicho fichero todos los experimentos hechos, mostrados para que el usuario pueda consultarlos y guardarlos.
- Por pantalla (Figura 11): Se muestra por pantalla dentro del entorno de *Java* los resultados obtenidos.
- En un fichero completo donde aparezcan todos los resultados, almacenados en filas según el incremento de las variables q_1 y q_2 (Figura 12).

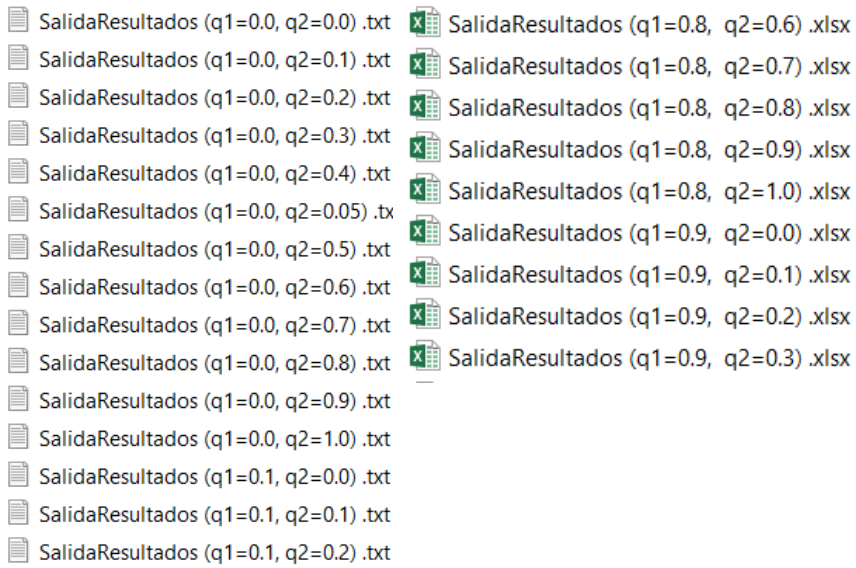


Figura 9. Formato de salida desglosado

Donde, dentro de cada fichero (según sea .txt o .xlsx):

```
Error Acumulado método Right-Shoulder: 1,1037
Error Acumulado método Linear: 3,2707
Error Acumulado método Power: 3,5765

Error MAPE Right-Shoulder (%): 9,1974
Error MAPE Linear (%): 27,2556
Error MAPE Power (%): 29,8044
```

Error Acumulado método Right-Shoulder	Error MAPE Right-Shoulder (%)
1,1037	9,1974 %
Error Acumulado método Linear	**Error MAPE Linear (%)**
3,2707	27,2556 %
Error Acumulado método Power	**Error MAPE Power (%)**
3,5765	29,8044 %

Figura 10. Contenido de un formato desglosado (txt y xlsx)

```
***************
OWA Brute Force Table

Tiempo de proceso: 0.028 segundos.

Brute-force     Right-shoulder  5,2896  ,1660 ,5130     [ ,0019 ,4803 ,4803 ,0375 ,0000 ,0000 ]
Brute-force     Linear  12,6139 ,0836 ,0082     [ ,0981 ,1804 ,1804 ,1804 ,1804 ,1804 ]
Brute-force     Power   13,4133 1,2431  [ ,1078 ,1474 ,1673 ,1816 ,1931 ,2028 ]

Tiempo de proceso: 53.329 segundos.

Monte Carlo     Right-shoulder  5,2954  ,1660 ,5182     [ ,0020 ,4732 ,4732 ,0517 ,0000 ,0000 ]
Monte Carlo     Linear  12,6139 ,1637 ,0949     [ ,0981 ,1804 ,1804 ,1804 ,1804 ,1804 ]
Monte Carlo     Power   16,0600 19,9999 [ ,0000 ,0000 ,0000 ,0003 ,0258 ,9739 ]

Tiempo de proceso: 0.602 segundos.
```

Figura 11. Formato de salida por pantalla

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	13	13	13	11	10	10	11	12	14	16	19
0,1		13	13	10	9	10	12	13	15	18	20
0,2			13	6	6	9	11	13	15	18	21
0,3				6	6	11	13	15	17	21	24
0,4					6	17	17	19	20	25	28
0,5						17	17	20	22	27	31
0,6							17	25	25	31	36
0,7								25	25	36	42
0,8									25	66	66
0,9										66	66
1											66

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	19	17	15	14	12	11	10	11	11	12	13
0,1	20	19	17	15	13	11	10	10	11	12	13
0,2	21	20	19	17	15	13	11	10	11	12	13
0,3	24	22	20	19	17	15	13	11	10	10	11
0,4	28	25	23	21	19	16	14	12	10	9	10
0,5	31	27	25	23	21	19	16	14	13	11	10
0,6	36	30	28	26	23	21	19	16	14	12	11
0,7	42	35	32	29	26	23	21	19	16	14	12
0,8	66	50	42	36	31	28	24	21	19	16	14
0,9	66	51	43	38	33	30	26	24	21	19	16
1	66	53	45	39	35	31	28	26	23	21	19

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	12	11	10	11	13	14	15	16	18
1	19	19	20	21	22	22	23	23	24	24
2	25	25	26	26	27	28	28	29	30	31
3	31	32	33	33	34	34	35	35	36	36
4	37	37	38	38	38	39	39	40	40	40
5	41	41	42	42	42	43	43	43	44	44
6	44	45	45	45	46	46	46	47	47	47
7	47	48	48	48	49	49	49	49	50	50
8	50	50	51	51	51	51	52	52	52	52
9	53	53	53	53	54	54	54	54	54	55
10	55	55	55	55	56	56	56	56	56	57

Figura 12. Formato de salida en fichero

3.4 Toma de decisiones

Se puede distinguir distintas tomas de decisiones a lo largo del diseño del proyecto, las cuales se van a plasmar en forma de respuesta a preguntas:

- ¿Qué forma de salida es más representativa?

  La más representativa es la tercera (en fichero), puesto que recoge en forma de tabla los resultados obtenidos. Es decir, muestra la distribución del MAPE para diferentes parámetros.

  El formato por pantalla también cumple, debido a que muestra el valor del mejor MAPE encontrado.

  Y, en cuanto al desglose, ocupa muchísimo tiempo. El tener que crear fichero, leer el original, calcular resultados, escribirlos en el fichero creado, … todo para cada pareja de q_1 y q_2 (además de si se quiere tanto en .txt como en .xlsx) es un coste computacional muy grande, además de que luego se ha de calcular el mejor MAPE. Mejor optar por otras vías (la mencionada anteriormente) para optimizar tiempo.

- ¿Por qué en el dataset número 1 se opta por datos europeos y no mundiales?

  No hay coincidencia en datos, puesto que por ejemplo en países de otros hemisferios puede no coincidir la temporada meteorológica. Es por eso por lo que se analizan los experimentos 1 y 2 por separado (con y sin la contribución de Buenos Aires). Cuando en países europeos es verano, allí es invierno, etc. Por ello, si se cuentan ciudades mundiales va a haber un margen de error demasiado grande. La temperatura, que es de la que se trata en esta primera aplicación, hay una restricción en la entrada según la procedencia del lugar a clasificar.

- ¿Por qué del dataset número 2 habiendo ya analizado el dataset número 1?

  Se debe a que el número de filas de datos es muy reducido en el primer caso. Es decir, solamente hay 12 filas (Enero, …, Diciembre). Por ello, en el dataset número 2 se ponen a disposición varios años, siendo mayor el número de datos a clasificar. Además, se pensó incluir datos horarios (clasificar datos ordenados las 24 horas), pero finalmente se ha optado por continuar con datos mensuales incluyendo varios años, puesto que así se ocupa un rango temporal mayor.

- ¿Por qué del dataset número 1 si se puede analizar el dataset número 2?
  Porque se da importancia también a la agregación de valores globales. No quedarse con un conjunto reducido, sino una ampliación de modo que puedan participar ciudades europeas del hemisferio norte

- ¿Posibilidad del análisis de precipitaciones u otras condiciones meteorológicas?
  Puede haber posibilidad, pero son variables más cambiantes. Es decir, hay una gran variación de datos (error grande) entre una ciudad y otra (por ejemplo precipitaciones en Bilbao y Murcia). Además, la medida de l/m^2 es relativa, porque la mayoría de días vale cero (no llueve) o próximo a cero. Tomando temperaturas, hay menor diferencia entre una ciudad y otra (aunque aún la hay).

- ¿Qué ciudad se predece y qué ciudades se dispone a clasificar? ¿Posibles alternativas?
  Es indispensable tener especial atención en qué ciudad se toma como salida y cuáles como entradas. No es lo mismo tener como salida la ciudad más cálida o la más fría, que tener una dentro de un rango intermedio entre todas las ciudades de entrada.
  Aún así, se pueden dar casos en los que se quiera tomar otra clasificación. Es por eso por lo que, para aplicaciones precisas, se podría clasificar con la totalidad de ciudades, o bien con sensores únicamente dentro de la comunidad autónoma en la que se encuentre la población.

- ¿Por qué se normalizan los datos de la tercera aplicación?
  En los datasets anteriores se tenían temperaturas, todas las entradas dentro del mismo rango. Pero en dataset número 3, las dos entradas se miden en diferentes niveles (millones de visitantes / miles de noches / miles de libras gastadas). Por ello, lo que se debe hacer es normalizar cada uno de los valores de cada entrada.

  Hay muchos métodos de normalización, pero lo estándar para pasar al intervalo de [0, 1] es dividir entre el máximo. Hay maneras más sofisticadas, pero cuando se quiere conseguir algo más. Por ejemplo, que los datos sigan una distribución normal, y ya se involucran media y desviación estándar. En este caso se piensa en dividir entre el máximo dejando un margen de 5%, que puede ser variable, es correcto: *ResultadoNormalizado = Cálculo / (MáximoValor * 1.05)*.

4. Implementación y experimentación

En esta etapa del proyecto se desarrollan las ideas expuestas en la etapa de diseño. Además, se describen los experimentos realizados, con una evaluación de las prestaciones de los mismos.

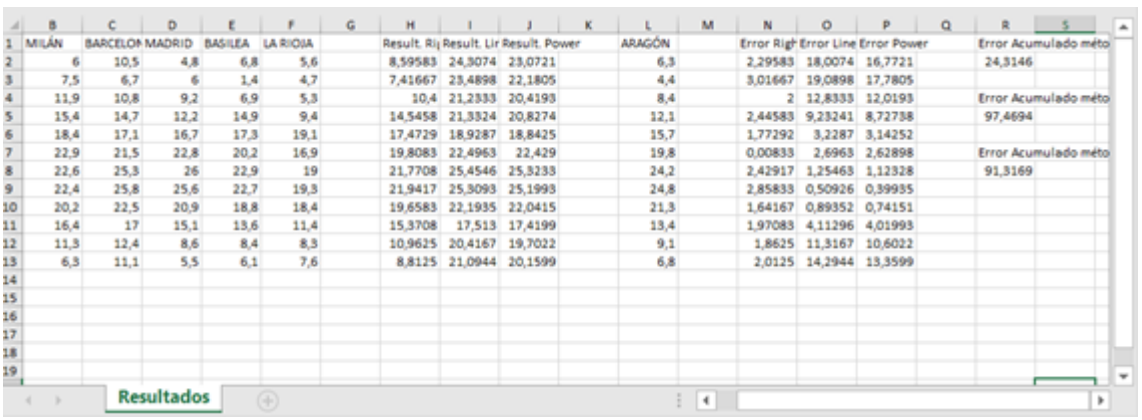
4.1 Proceso de implementación

En este apartado, se pretende plasmar el proceso de elaboración de un proyecto en *Java* que satisfaga las necesidades planteadas en secciones anteriores. Para ello, se distinguen tres fases:

- Procesamiento.
- Se asegura la lectura y escritura de ficheros, tanto *.txt* como *.xlsx*.
- Se adecuan los datasets de entradas de forma que tenga varias columnas de entrada, y una de salida. De esta forma, se "unifican", siendo el mismo algoritmo será el que se usará para una aplicación u otra.

- Cálculos.
- Se introducen los métodos de cálculo. Es decir, el cálculo de los errores, tanto individualmente como el error total. Responde a la forma de txt y xlsx:

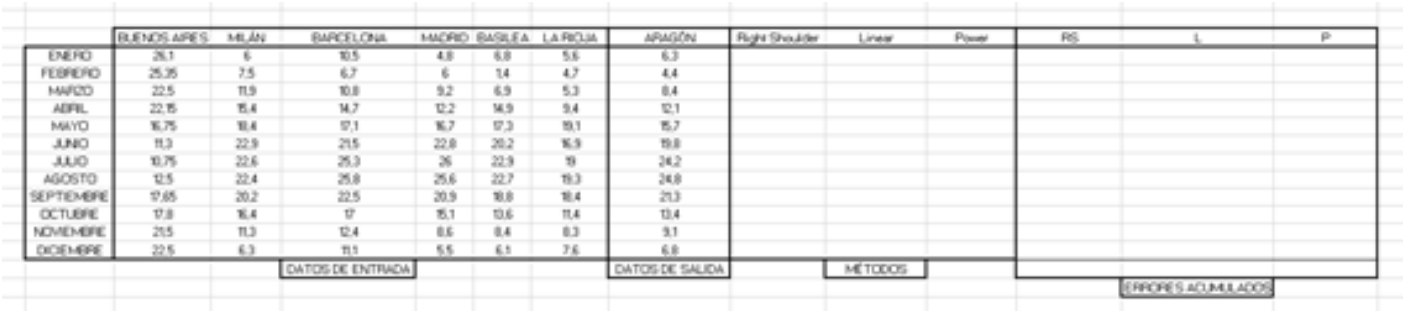
```
19.089814814814815
12.833333333333334
9.232407407407406
3.2287037037037045
2.696296296296296
1.2546296296296333
0.5092592592592631
0.893518518518519
4.112962962962962
11.316666666666668
14.294444444444444

Error con método Power
16.772082243022496
17.780477377123745
12.019271617431022
8.727381519007713
3.1425212930921305
2.6289826691596616
1.123281183021188
0.3993458468917197
0.741506777103087
4.019934455789189
10.602214686498078
13.359940569024587

Error Acumulado método Right-Shoulder: 24.31458333333334
Error Acumulado método Linear: 97.46944444444446
Error Acumulado método Power: 91.3169402371646
```



	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	MILÁN	BARCELON	MADRID	BASILEA	LA RIOJA		Result. Ri	Result. Lir	Result. Power		ARAGÓN		Error Righ	Error Line	Error Power		Error Acumulado méto	
2	6	10,5	4,8	6,8	5,6		8,59583	24,3074	23,0721		6,3		2,29583	18,0074	16,7721		24,3146	
3	7,5	6,7	6	1,4	4,7		7,41667	23,4898	22,1805		4,4		3,01667	19,0898	17,7805			
4	11,9	10,8	9,2	6,9	5,3		10,4	21,2333	20,4193		8,4		2	12,8333	12,0193		Error Acumulado méto	
5	15,4	14,7	12,2	14,9	9,4		14,5458	21,3324	20,8274		12,1		2,44583	9,23241	8,72738		97,4694	
6	18,4	17,1	16,7	17,3	19,1		17,4729	18,9287	18,8425		15,7		1,77292	3,2287	3,14252			
7	22,9	21,5	22,8	20,2	16,9		19,8083	22,4963	22,429		19,8		0,00833	2,6963	2,62898		Error Acumulado méto	
8	22,6	25,3	26	22,9	19		21,7708	25,4546	25,3233		24,2		2,42917	1,25463	1,12328		91,3169	
9	22,4	25,8	25,6	22,7	19,3		21,9417	25,3093	25,1993		24,8		2,85833	0,50926	0,39935			
10	20,2	22,5	20,9	18,8	18,4		19,6583	22,1935	22,0415		21,3		1,64167	0,89352	0,74151			
11	16,4	17	15,1	13,6	11,4		15,3708	17,513	17,4199		13,4		1,97083	4,11296	4,01993			
12	11,3	12,4	8,6	8,4	8,3		10,9625	20,4167	19,7022		9,1		1,8625	11,3167	10,6022			
13	6,3	11,1	5,5	6,1	7,6		8,8125	21,0944	20,1599		6,8		2,0125	14,2944	13,3599			
14																		
15																		
16																		
17																		
18																		
19																		

Resultados

- Se adapta el fichero de salida, de forma que sea de la forma:



	BUENOS AIRES	MILÁN	BARCELONA	MADRID	BASILEA	LA RIOJA	ARAGÓN	Right Shoulder	Linear	Power	RS	L	P
ENERO	26.1	6	10.5	4.8	6.8	5.6	6.3						
FEBRERO	25.35	7.5	6.7	6	1.4	4.7	4.4						
MARZO	22.5	11.9	10.8	9.2	6.9	5.3	8.4						
ABRIL	22.15	15.4	14.7	12.2	14.9	9.4	12.1						
MAYO	16.75	18.4	17.1	16.7	17.3	19.1	15.7						
JUNIO	11.3	22.9	21.5	22.8	20.2	16.9	19.8						
JULIO	10.75	22.6	25.3	26	22.9	19	24.2						
AGOSTO	12.5	22.4	25.8	25.6	22.7	19.3	24.8						
SEPTIEMBRE	17.65	20.2	22.5	20.9	18.8	18.4	21.3						
OCTUBRE	17.8	16.4	17	15.1	13.6	11.4	13.4						
NOVIEMBRE	21.5	11.3	12.4	8.6	8.4	8.3	9.1						
DICIEMBRE	22.5	6.3	11.1	5.5	6.1	7.6	8.8						
			DATOS DE ENTRADA				DATOS DE SALIDA		MÉTODOS				
												ERRORES ACUMULADOS	

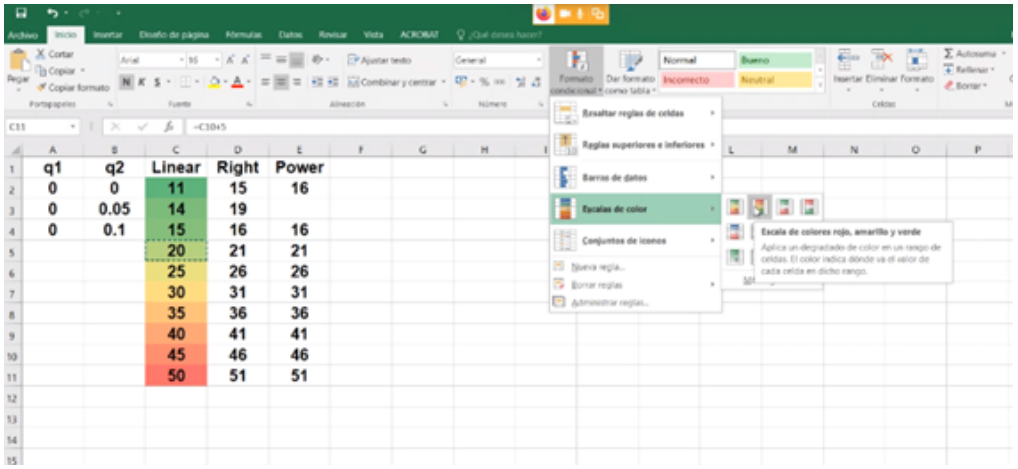
Nótese que es un ejemplo, correspondiente a la aplicación 1 de este TFG.

- Una vez obtenida la lectura, escritura y cálculo dentro de un formato adecuado, se procede a la realización de dichos cálculos, sacándolos igualmente por fichero, dentro de un bucle para que los valores de q_1 y q_2 vayan cambiando (no se introduzcan por teclado, sino que se vayan actualizando de manera continua), de forma que muestre los resultados de una de las siguientes maneras:

```
Método    q1    q2    Error total

Right     0.01    0.01    99.2

Right     0.01    0.02    98.1

Right     0.01    0.03    98.5

...
```

```
Método    q1    q2    Error total
Right     0.01    0.01    99.2
Linear     0.01    0.01     98.5
Power     0.01    0.01     98.1
Right     0.01    0.02    99.1
...
```

- Depuración.
- Se distinguen tres métodos (Right-Shoulder, Linear, Power), donde cada uno es llamado dependiendo del caso, tal como se había comentado en la sección 3.1.
- Como comparativa final, los resultados deben aparecer en una salida de tipo tabla, donde se represente mejor gráficamente. Podrá verse de manera más intuitiva.



	A	B	C	D	E
1	q1	q2	Linear	Right	Power
2	0	0	11	15	16
3	0	0.05	14	19	
4	0	0.1	15	16	16
5			20	21	21
6			25	26	26
7			30	31	31
8			35	36	36
9			40	41	41
10			45	46	46
11			50	51	51

- Se recurre al “DecimalFormat”, importando dicha clase para mostrar el error en menos decimales. Además, sirve también de conversión de “.” a “,”, de manera que luego pueda utilizarse para abrir en *Excel* y comparar datos entre sí.
- Se da la opción al usuario para que seleccione sobre qué dataset quiere trabajar. Es decir, qué aplicación quiere ejecutar. Quedaría, visualmente, como:

```
Introduzca sobre qué dataset quiere trabajar:
        * Introduzca '1' si quiere trabajar con datos de temperaturas europeas del 2018
        * Introduzca '2' si quiere trabajar con datos de temperaturas españolas entre 2014-2018
        * Introduzca '3' si quiere trabajar con datos de turismo de Londres entre 2002-2019
```

- Se añade una medida del tiempo de proceso, la cual se hace de la forma:

```
long tiempoInicio = System.currentTimeMillis();
...Código de la tarea que se quiere medir...
long tiempoTotal = System.currentTimeMillis() - tiempoInicio;
System.out.println("Segundos: " + tiempoTotal/1000.);
```

- Se introduce el método de Monte Carlo, a modo de comparativa de resultados.
- Se adapta la salida por pantalla, de forma que sea:

```
Brute-force    Right    9,0458    ,3180 ,8330       ,0000 ,0298 ,3236 ,3236 ,3230
,0000        2,748
Brute-force    Linear   9,2382    ,2670 ,0000       ,0000 ,0905 ,2274 ,2274 ,2274
,2274        2,748
Brute-force    Power    10,6287   2,2010            ,0194 ,0697 ,1284 ,1922 ,2598
,3305        2,748
```

en vez de:

```
OWA Brute Force
Right-shoulder
q1:    ,3180    q2:    ,8330    MAPE:    9,0458
W:    ,0000    ,0298    ,3236    ,3236    ,3230    ,0000
Linear
q1:    ,2670    q2:    ,0000    MAPE:    9,2382
W:    ,0000    ,0905    ,2274    ,2274    ,2274    ,2274
Power
q:    2,2010            MAPE:    10,6287
W:    ,0194    ,0697    ,1284    ,1922    ,2598    ,3305
Tiempo: 2.748 segundos
```

- Revisión general del algoritmo.

El código fuente resultante final se encuentra en el Anexo II (sección 6.2).

4.2 Evaluación de los experimentos

Esta sección hace referencia a la evaluación de los resultados obtenidos para cada una de las aplicaciones que se exponen en este TFG.

4.2.1 Consideraciones previas

El código está implementado en Java 1.8. Todos los experimentos siendo ejecutados en un ordenador portátil con Intel Core i7-8750H, 16GB RAM, 1TB HDD + 256GB SSD, sobre Windows 10, 64-bit.

Además, se han considerado ciertos valores de parámetros, que van a condicionar el resultado final, así como las prestaciones temporales de cada aplicación:

- Incremento "Delta" para el algoritmo (con el que van a aumentar los valores de q_i) de $\Delta = 0{,}001$ y $\Delta = 0{,}0001$.
- Para la función del cuantificador "Power", se toma como $q = 20$ el límite superior. Es decir, $q \in (0, 20]$. Se ha propuesto esta selección tras comprobar que para valores mayores se producen unos cambios muy pequeños en un vector de 5 pesos.
- El número de repeticiones a realizar en el algoritmo "Monte Carlo" es de 5·10^5 veces. Este valor se ha escogido para tener un tiempo de ejecución bajo, y similar al algoritmo de "Force Brute" con $\Delta = 0{,}001$. Además, tras ejecutarlo varias veces, el resultado no variaba entre una ejecución y otra, en términos de redondeo a dos decimales.

De cara a la evaluación de cada aplicación, que se va a desarrollar a continuación, se plasma tanto la representación de salida por pantalla como la representación de la variación del MAPE (formas 2 y 3 de los "Formatos de salida" de la sección 3.3).

4.2.2 Primera aplicación

Corresponde al dataset número 1 (temperaturas mundiales en el año 2018), que se compone de dos experimentos:

- Experimento 1: Predecir la temperatura de Zaragoza a partir del resto, sin contar con la aportación de la entrada de Buenos Aires (Figuras 13, 14).
- Experimento 2: Predecir la temperatura de Zaragoza a partir del resto, contando con la aportación de la entrada de Buenos Aires (Figuras 15, 16).

Se observa (Figuras 14 y 16) que la distribución del MAPE en estos experimentos es no homogénea. En el experimento 1, se distribuye de tal forma que:
- Para la función Right-Shoulder, los valores de dicho error cuyo q_1 es bajo, es pequeño, aumentando progresivamente con el valor de dicho parámetro.
- En la función Linear, el experimento da mejores resultados (menor error) con valores altos de q_2.
- En la Power, el error es en general alto, pero se puede destacar un margen donde es bajo, correspondiente a cuando el valor de q es mínimo.

En el experimento 2, de la misma manera, pero el error es visiblemente mayor en los tres tipos de cuantificadores. Esto es debido a que la contribución de Buenos Aires empeora los resultados. Los valores correspondientes al dataset difieren mucho respecto al resto de entradas, debido a que, por ejemplo, los meses de invierno en Europa se corresponden a los de verano en Buenos Aires.
Esto se traduce en que, de forma general el error aumenta, destacando:
- Right-Shoulder: cuando q_1 es mínimo, y q_1 es bajo, el error MAPE es alto.
- Linear: valores de q_1 bajos con q_1 altos, dan lugar a MAPE alto.
- Power: manera de distribución similar al experimento 1, pero con valores del MAPE mucho mayores para $q = 0$.

El mínimo obtenido (Figuras 13 y 15) es similar para ambos experimentos para Right-Shoulder, pero notablemente mejores en el primero para Linear y Power.

En cuanto al tiempo, es mayor para el 2, puesto que hay una entrada más.

```
***************
OWA Brute Force Table

Tiempo de proceso: 1.873 segundos.

Brute-force     Right-shoulder  5,3888  ,1991 ,3851     [ ,0048 ,9952 ,0000 ,0000 ,0000 ]
Brute-force     Linear  9,4604  ,4434 ,9319     [ ,4203 ,4203 ,1104 ,0245 ,0245 ]
Brute-force     Power   10,0735 ,3014   [ ,6156 ,1430 ,0986 ,0777 ,0650 ]

Tiempo de proceso BruteForce (0.0001): 46.748 segundos.

Brute-force     Right-shoulder  5,3956  ,1990 ,4000     [ ,0050 ,9950 ,0000 ,0000 ,0000 ]
Brute-force     Linear  9,4614  ,4430 ,9320     [ ,4208 ,4208 ,1096 ,0244 ,0244 ]
Brute-force     Power   10,0741 ,3010   [ ,6160 ,1429 ,0985 ,0776 ,0650 ]

Tiempo de proceso BruteForce (0.001): 0.601 segundos.

Monte Carlo     Right-shoulder  5,3890  ,1991 ,3787     [ ,0048 ,9952 ,0000 ,0000 ,0000 ]
Monte Carlo     Linear  9,4613  ,4421 ,9309     [ ,4212 ,4212 ,1081 ,0248 ,0248 ]
Monte Carlo     Power   10,0735 6,0285  [ ,0001 ,0039 ,0420 ,2145 ,7395 ]

Tiempo de proceso Montecarlo (500000): 0.71 segundos.
```

Figura 13. Resultados E1

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	13,1	13,1	13,1	10,9	9,88	9,7	10,7	12,4	13,8	16,4	19
0,1		13,1	13,1	9,88	9,3	10,2	11,6	13,5	14,9	17,6	20
0,2			13,1	5,72	5,72	8,85	10,7	13,1	14,9	18,4	21
0,3				5,72	5,72	10,7	12,6	15,3	17	20,9	24
0,4					5,72	16,8	16,8	19,2	20,5	24,8	28
0,5						16,8	16,8	20,5	21,8	27,1	31
0,6							16,8	24,6	24,6	31,3	36
0,7								24,6	24,6	35,9	42
0,8									24,6	66	66
0,9										66	66
1											66

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	18,6	17	15,4	13,8	12,1	10,7	10,4	10,7	11,3	12,1	13
0,1	19,9	18,6	16,8	15	13,2	11,4	10,4	10,5	11,2	12	13
0,2	21,3	20	18,6	16,6	14,6	12,5	10,7	10,4	11	11,8	13
0,3	24,1	21,8	20,4	18,6	16,6	14,6	12,7	10,8	9,84	10,1	11
0,4	28,2	24,6	22,8	20,7	18,6	16,4	14,3	12,3	10,4	9,5	9,9
0,5	31	26,7	24,9	22,8	20,7	18,6	16,5	14,5	12,6	10,7	9,7
0,6	35,9	30,3	28	25,6	23,1	20,8	18,6	16,4	14,4	12,4	11
0,7	41,8	34,7	31,7	28,7	25,9	23,3	20,9	18,6	16,4	14,3	12
0,8	66	50,1	41,8	36,1	31,5	27,6	24,2	21,2	18,6	16,1	14
0,9	66	51,5	43,3	37,8	33,4	29,7	26,4	23,5	20,9	18,6	16
1	66	52,7	44,6	39,3	35,1	31,5	28,3	25,5	23	20,7	19

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	11,5	10,7	10,1	10,9	12,5	13,9	15,3	16,5	17,6
1	18,6	19,5	20,3	21	21,7	22,3	22,9	23,5	24	24,5
2	25	25,4	25,9	26,3	26,8	27,6	28,5	29,2	30	30,6
3	31,3	31,9	32,6	33,1	33,7	34,3	34,8	35,3	35,8	36,3
4	36,7	37,2	37,6	38,1	38,5	38,9	39,3	39,7	40,1	40,5
5	40,9	41,2	41,6	42	42,3	42,7	43	43,4	43,7	44,1
6	44,4	44,7	45	45,3	45,7	46	46,3	46,6	46,9	47,2
7	47,5	47,8	48,1	48,3	48,6	48,9	49,2	49,4	49,7	50
8	50,2	50,5	50,7	51	51,2	51,5	51,7	52	52,2	52,4
9	52,7	52,9	53,1	53,3	53,6	53,8	54	54,2	54,4	54,6
10	54,8	55	55,2	55,4	55,6	55,8	56	56,2	56,3	56,5
11	56,7	56,9	57	57,2	57,4	57,5	57,7	57,8	58	58,1
12	58,3	58,4	58,6	58,7	58,9	59	59,1	59,3	59,4	59,5
13	59,7	59,8	59,9	60	60,1	60,3	60,4	60,5	60,6	60,7
14	60,8	60,9	61	61,1	61,2	61,3	61,4	61,5	61,6	61,7
15	61,8	61,8	61,9	62	62,1	62,2	62,3	62,3	62,4	62,5
16	62,6	62,6	62,7	62,8	62,8	62,9	63	63	63,1	63,1
17	63,2	63,3	63,3	63,4	63,4	63,5	63,5	63,6	63,6	63,7
18	63,7	63,8	63,8	63,9	63,9	64	64	64,1	64,1	64,1
19	64,2	64,2	64,3	64,3	64,3	64,4	64,4	64,4	64,5	64,5
20	64,5	64,6	64,6	64,6	64,7	64,7	64,7	64,7	64,8	64,8

Figura 14. MAPE E1

```
***************
OWA Brute Force Table

Tiempo de proceso: 1.934 segundos.

Brute-force     Right-shoulder  5,2896  ,1660 ,5130     [ ,0019 ,4803 ,4803 ,0375 ,0000 ,0000 ]
Brute-force     Linear  12,6139 ,0836 ,0082     [ ,0981 ,1804 ,1804 ,1804 ,1804 ,1804 ]
Brute-force     Power   13,4133 1,2431  [ ,1078 ,1474 ,1673 ,1816 ,1931 ,2028 ]

Tiempo de proceso BruteForce (0.0001): 51.627 segundos.

Brute-force     Right-shoulder  5,2896  ,1660 ,5130     [ ,0019 ,4803 ,4803 ,0375 ,0000 ,0000 ]
Brute-force     Linear  12,6139 ,1250 ,0530     [ ,0981 ,1804 ,1804 ,1804 ,1804 ,1804 ]
Brute-force     Power   13,4136 1,2430  [ ,1078 ,1474 ,1673 ,1816 ,1931 ,2028 ]

Tiempo de proceso BruteForce (0.001): 0.507 segundos.

Monte Carlo     Right-shoulder  5,2974  ,1660 ,5193     [ ,0020 ,4717 ,4717 ,0547 ,0000 ,0000 ]
Monte Carlo     Linear  12,6139 ,1438 ,0733     [ ,0981 ,1804 ,1804 ,1804 ,1804 ,1804 ]
Monte Carlo     Power   16,0601 19,9999 [ ,0000 ,0000 ,0000 ,0003 ,0258 ,9739 ]

Tiempo de proceso Montecarlo (500000): 0.609 segundos.
```

Figura 15. Resultados E2

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	35,7	36	34	29,4	26,3	23,9	21,6	19,6	17,7	16,61	16
0,1		36	31	24,3	20,2	17,4	14,7	12,5	11	10,88	13
0,2			7,2	7,22	5,87	5,49	6,34	8,54	10,9	15,28	20
0,3				7,22	5,79	6,71	9,56	12	14,4	19,21	25
0,4					8,3	8,3	12,6	15	17,5	22,96	29
0,5						8,3	18	19,2	21,2	27,56	35
0,6							18	20,4	23	31,22	40
0,7								25,7	25,7	37,75	49
0,8									25,7	55,99	70
0,9										92,68	93
1											93

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	16,1	21	24	26,6	28,7	30,4	31,8	32,9	33,9	34,73	36
0,1	12,9	16	21	24,6	27,4	29,5	31,2	32,6	33,7	34,63	36
0,2	20,1	13	16	20,6	24	26,6	28,6	30,2	31,6	32,62	34
0,3	24,6	17	13	16,1	19,3	22	24,1	25,8	27,2	28,32	29
0,4	29	21	16	13,9	16,1	18,6	20,8	22,6	24	25,26	26
0,5	34,5	25	20	15,4	14,3	16,1	18,2	20	21,5	22,83	24
0,6	39,8	30	24	19	15,3	14,7	16,1	17,7	19,3	20,54	22
0,7	48,9	37	30	24,1	19,1	15,5	15,1	16,1	17,4	18,56	20
0,8	70,2	52	41	32,8	26,1	20,4	16,1	15,6	16,1	16,87	18
0,9	92,7	66	51	40,8	32,7	26,1	20,4	16,3	15,8	16,06	17
1	92,7	68	53	43,5	35,7	29,2	23,7	18,8	16	15,84	16

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	34	32	30,2	28,2	26,2	24,2	22,1	20	17,94
1	16,1	15	14	13,8	14,7	16,4	17,9	19,3	20,7	21,95
2	23,2	24	25	26,8	28,1	29,4	30,7	31,8	32,9	33,96
3	34,9	36	37	37,7	38,5	39,3	40,1	40,8	41,6	42,28
4	43	44	44	45	45,6	46,2	46,8	47,4	48	48,58
5	49,1	50	50	50,8	51,3	51,9	52,4	52,9	53,4	53,91
6	54,4	55	55	55,9	56,3	56,8	57,3	57,7	58,2	58,63
7	59,1	60	60	60,4	60,8	61,2	61,6	62	62,4	62,85
8	63,3	64	64	64,4	64,8	65,2	65,6	65,9	66,3	66,65
9	67	67	68	68,1	68,4	68,7	69,1	69,4	69,7	70,06
10	70,4	71	71	71,3	71,6	71,9	72,2	72,5	72,8	73,11
11	73,4	74	74	74,2	74,5	74,8	75	75,3	75,6	75,82
12	76,1	76	77	76,8	77	77,3	77,5	77,8	78	78,21
13	78,4	79	79	79,1	79,3	79,5	79,7	79,9	80,1	80,3
14	80,5	81	81	81,1	81,3	81,4	81,6	81,8	82	82,13
15	82,3	82	83	82,8	83	83,1	83,3	83,4	83,6	83,72
16	83,9	84	84	84,3	84,4	84,6	84,7	84,8	85	85,09
17	85,2	85	85	85,6	85,7	85,8	85,9	86	86,2	86,26
18	86,4	86	87	86,7	86,8	86,9	87	87,1	87,2	87,27
19	87,4	87	88	87,6	87,7	87,8	87,9	88	88	88,12
20	88,2	88	88	88,4	88,5	88,6	88,6	88,7	88,8	88,85

Figura 16. MAPE E2

4.2.3 Segunda aplicación

Se trata del dataset número 2 (temperaturas españolas entre los años 2014 y 2018), que se da en un experimento:

- Experimento 3: Predicción de la temperatura de Zaragoza a partir de las temperaturas del resto (Figuras 17, 18).

  Se observa (Figura 18) que la distribución del MAPE en estos experimentos es más homogénea que la de la aplicación anterior. Se distribuye de tal forma que:
  - Right-Shoulder: error bajo, pero mayor para los casos de q_1 y q_2 bajos, y para q_1 y q_2 altos.
  - Linear: peor caso para q_1 bajo y q_2 alto, o para q_1 alto y q_2 bajo.
  - Power: peor caso para valor de q mínimo (q = 0) o para q alto.

  El mínimo obtenido (Figura 17) es relativamente bajo, menor que lo obtenido en la primera aplicación, para las funciones Linear y Power. Sin embargo, para la función Right-Shoulder, el resultado es peor.

  En cuanto al tiempo, se observa un aumento considerable de dicho tiempo de ejecución, de forma que se encuentra en los 228 segundos (para los alrededor de 50 de la primera aplicación). Esto se debe a que, pese a que el número de entradas es el mismo, hay mayor cantidad de datos (se dispone de cinco veces la longitud de la aplicación anterior. Es por eso por lo que, pese a que pueda dar mejores o peores resultados, el tiempo transcurrido hasta que se evalúan todos los datos es mayor.

```
***************
OWA Brute Force Table

Tiempo de proceso: 3.904 segundos.

Brute-force     Right-shoulder  8,7904  ,5605 ,8303     [ ,0000 ,0000 ,0000 ,3935 ,6065 ,0000 ]
Brute-force     Linear  9,5211  ,6666 ,4382     [ ,1096 ,1096 ,1096 ,1096 ,2808 ,2808 ]
Brute-force     Power   9,7050  1,5976  [ ,0571 ,1158 ,1575 ,1928 ,2241 ,2527 ]

Tiempo de proceso BruteForce (0.0001): 228.182 segundos.

Brute-force     Right-shoulder  8,7904  ,5860 ,7910     [ ,0000 ,0000 ,0000 ,3935 ,6065 ,0000 ]
Brute-force     Linear  9,5216  ,6660 ,4370     [ ,1094 ,1094 ,1094 ,1100 ,2809 ,2809 ]
Brute-force     Power   9,7050  1,5980  [ ,0571 ,1157 ,1575 ,1928 ,2241 ,2527 ]

Tiempo de proceso BruteForce (0.001): 2.267 segundos.

Monte Carlo     Right-shoulder  8,7904  ,6428 ,7035     [ ,0000 ,0000 ,0000 ,3935 ,6065 ,0000 ]
Monte Carlo     Linear  9,5220  ,6670 ,4388     [ ,1096 ,1096 ,1096 ,1096 ,2805 ,2809 ]
Monte Carlo     Power   10,8831 20,0000 [ ,0000 ,0000 ,0000 ,0003 ,0258 ,9739 ]

Tiempo de proceso Montecarlo (500000): 2.356 segundos.
```

Figura 17. Resultados E3

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	22	22	21,2	19,7	18,2	17,1	15,9	14,6	13	11,5	11
0,1		22	20,3	18,4	16,9	15,8	14,6	13,3	11,8	10,6	10
0,2			16,5	16,5	15,1	14,2	13,1	12,1	10,7	9,82	9,7
0,3				16,5	13,8	13,2	12,2	11,3	9,93	9,43	9,7
0,4					12,8	12,8	11,7	10,7	9,29	9,32	9,8
0,5						12,8	10,9	9,97	8,88	9,38	11
0,6							10,9	9,18	8,71	10,4	13
0,7								10,8	10,8	14,7	18
0,8									10,8	20,2	23
0,9										26,8	27
1											27

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	10,7	11,4	12,3	13,4	14,5	15,8	17,1	18,4	19,7	20,9	22
0,1	9,95	10,7	11,5	12,5	13,7	15,1	16,6	18	19,4	20,8	22
0,2	9,74	9,94	10,7	11,5	12,6	13,9	15,3	16,9	18,4	19,8	21
0,3	9,74	9,63	9,97	10,7	11,5	12,6	13,9	15,3	16,8	18,3	20
0,4	9,81	9,65	9,58	9,98	10,7	11,5	12,6	13,8	15,3	16,7	18
0,5	10,5	9,83	9,59	9,56	9,97	10,7	11,5	12,6	14	15,5	17
0,6	13,3	10,9	9,92	9,55	9,53	9,95	10,7	11,6	12,8	14,2	16
0,7	18,3	14,7	11,8	10,2	9,64	9,57	9,95	10,7	11,6	12,9	15
0,8	23,3	19,2	15,5	12,6	10,7	10,1	9,88	10,1	10,7	11,6	13
0,9	26,8	22,5	18,5	15,1	12,5	10,8	10,2	10	10,2	10,7	12
1	26,8	22,9	19,3	16,1	13,5	11,4	10,5	10,1	10	10,2	11

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	20,2	18,5	16,9	15,4	14,2	13,2	12,3	11,6	11,1
1	10,7	10,3	10	9,83	9,69	9,65	9,64	9,67	9,72	9,77
2	9,81	9,84	9,92	10	10,1	10,2	10,3	10,5	10,7	10,9
3	11,1	11,4	11,6	11,9	12,2	12,5	12,8	13,1	13,4	13,6
4	13,9	14,2	14,5	14,8	15	15,3	15,5	15,8	16	16,3
5	16,5	16,7	17	17,2	17,4	17,6	17,8	18	18,2	18,3
6	18,5	18,7	18,9	19	19,2	19,3	19,5	19,6	19,8	19,9
7	20,1	20,2	20,3	20,5	20,6	20,7	20,8	21	21,1	21,2
8	21,3	21,4	21,5	21,6	21,7	21,8	21,9	22	22,1	22,2
9	22,3	22,4	22,5	22,5	22,6	22,7	22,8	22,9	22,9	23
10	23,1	23,2	23,2	23,3	23,4	23,4	23,5	23,5	23,6	23,7
11	23,7	23,8	23,8	23,9	24	24	24,1	24,1	24,2	24,2
12	24,3	24,3	24,4	24,4	24,5	24,5	24,5	24,6	24,6	24,7
13	24,7	24,7	24,8	24,8	24,9	24,9	24,9	25	25	25
14	25,1	25,1	25,1	25,2	25,2	25,2	25,3	25,3	25,3	25,3
15	25,4	25,4	25,4	25,5	25,5	25,5	25,5	25,6	25,6	25,6
16	25,6	25,6	25,7	25,7	25,7	25,7	25,7	25,8	25,8	25,8
17	25,8	25,8	25,9	25,9	25,9	25,9	25,9	25,9	26	26
18	26	26	26	26	26,1	26,1	26,1	26,1	26,1	26,1
19	26,1	26,1	26,2	26,2	26,2	26,2	26,2	26,2	26,2	26,2
20	26,3	26,3	26,3	26,3	26,3	26,3	26,3	26,3	26,3	26,3

Figura 18. MAPE E3

4.2.4 Tercera aplicación

Hace referencia al dataset número 3 (datos de turismo de la ciudad de Londres entre 2002 y 2019), que se subdivide en tres experimentos:

- Experimento 4: Predecir el número de visitas totales a partir del total de noches alojadas y del total gastado (Figuras 19, 20).
- Experimento 5: Predecir el número de noches de alojamiento de turistas a partir del número total de visitas y el total gastado (Figuras 21, 22).
- Experimento 6: Predecir el total gastado a partir del total de noches alojadas y del total de visitas (Figuras 23, 24).

Se aprecia un resultado de distribución del MAPE similar para los experimentos 4 y 5 (Figuras 20 y 22):
- Right-Shoulder: peor caso de error para valores de q_1 altos.
- Linear: peor caso para q_1 alto y q_2 bajo.
- Power: peor caso para valor de q alto.

Para el experimento 6, se comporta de forma contraria (Figura 24):
- Right-Shoulder: peor caso de error para valores de q_1 bajos.
- Linear: peor caso para q_1 bajo y q_2 alto.
- Power: peor caso para valor de q bajo.

La distribución general del error es considerablemente menor para el experimento 6, por lo que a priori se ven mejores resultados visibles. Pero, en la búsqueda del mínimo (Figuras 19 y 21), se observa que, pese a tener mejor distribución, el resultado del valor del mínimo MAPE es mayor que el obtenido en los experimentos 4 y 5.
Cabe destacar que el valor mínimo es idéntico para los tres tipos de cuantificadores, siendo este además igual en los experimentos 4 y 5.

En cuanto al tiempo de ejecución, es similar en los tres experimentos, puesto que el conjunto de datos a tener en cuenta en esta aplicación es el mismo en los tres experimentos, situándose en un valor intermedio entre la primera y la segunda aplicación.

```
***************
OWA Brute Force Table

Tiempo de proceso: 2.539 segundos.

Brute-force     Right-shoulder  9,0070  ,1492 ,5278     [ ,9266 ,0734 ]
Brute-force     Linear  9,0070  ,0535 ,8610     [ ,9266 ,0734 ]
Brute-force     Power   9,0070  ,1100   [ ,9266 ,0734 ]

Tiempo de proceso BruteForce (0.0001): 102.786 segundos.

Brute-force     Right-shoulder  9,0070  ,2350 ,5210     [ ,9266 ,0734 ]
Brute-force     Linear  9,0070  ,1420 ,8740     [ ,9266 ,0734 ]
Brute-force     Power   9,0070  ,1100   [ ,9266 ,0734 ]

Tiempo de proceso BruteForce (0.001): 1.066 segundos.

Monte Carlo     Right-shoulder  9,0070  ,0730 ,5338     [ ,9266 ,0734 ]
Monte Carlo     Linear  9,0070  ,0590 ,8618     [ ,9266 ,0734 ]
Monte Carlo     Power   9,0070  2,2005  [ ,2176 ,7824 ]

Tiempo de proceso Montecarlo (500000): 1.147 segundos.
```

Figura 19. Resultados E4

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	9,17	9,17	9,171	9,17	9,171	9,17	9,149	9,69	10,6	11,5	13
0,1		9,17	9,171	9,17	9,171	9,17	9,237	10,1	11,3	12,7	14
0,2			9,171	9,17	9,171	9,17	9,454	10,9	12,7	14,4	16
0,3				9,17	9,171	9,17	10,12	12,7	15,1	16,8	18
0,4					9,171	9,17	12,7	16,8	19,2	20,6	22
0,5						9,17	27,31	27,3	27,3	27,3	27
0,6							27,31	27,3	27,3	27,3	27
0,7								27,3	27,3	27,3	27
0,8									27,3	27,3	27
0,9										27,3	27
1											27

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	12,7	11,6	10,89	10,3	9,815	9,45	9,237	9,1	9,03	9,03	9,2
0,1	14	12,7	11,53	10,7	10,12	9,63	9,324	9,15	9,03	9,02	9,2
0,2	15,7	14,1	12,7	11,4	10,58	9,93	9,454	9,2	9,04	9,02	9,2
0,3	18,1	16,2	14,37	12,7	11,26	10,4	9,692	9,29	9,09	9,01	9,2
0,4	21,7	19,2	16,83	14,7	12,7	11,1	10,12	9,45	9,15	9,01	9,2
0,5	27,3	23,8	20,64	17,7	15,09	12,7	10,89	9,82	9,24	9,03	9,2
0,6	27,3	24,4	21,67	19,2	16,83	14,7	12,7	11,1	10,1	9,45	9,1
0,7	27,3	24,8	22,42	20,2	18,13	16,2	14,37	12,7	11,3	10,4	9,7
0,8	27,3	25,1	23	21	19,15	17,4	15,73	14,1	12,7	11,4	11
0,9	27,3	25,3	23,45	21,7	19,97	18,4	16,83	15,4	14	12,7	12
1	27,3	25,5	23,82	22,2	20,64	19,2	17,74	16,4	15,1	13,8	13

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	9,01	9,05	9,2	9,417	9,75	10,2	10,7	11,2	11,9
1	12,7	13,5	14,2	14,9	15,63	16,3	16,92	17,5	18,1	18,6
2	19,2	19,6	20,11	20,5	20,96	21,4	21,72	22,1	22,4	22,7
3	23	23,3	23,53	23,8	24	24,2	24,41	24,6	24,8	24,9
4	25,1	25,2	25,37	25,5	25,61	25,7	25,83	25,9	26	26,1
5	26,2	26,3	26,32	26,4	26,45	26,5	26,56	26,6	26,7	26,7
6	26,7	26,8	26,81	26,8	26,88	26,9	26,93	27	27	27
7	27	27	27,06	27,1	27,09	27,1	27,12	27,1	27,1	27,2
8	27,2	27,2	27,18	27,2	27,2	27,2	27,21	27,2	27,2	27,2
9	27,2	27,2	27,24	27,2	27,25	27,3	27,26	27,3	27,3	27,3
10	27,3	27,3	27,28	27,3	27,28	27,3	27,28	27,3	27,3	27,3
11	27,3	27,3	27,29	27,3	27,29	27,3	27,29	27,3	27,3	27,3
12	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3
13	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3	27,3
14	27,3	27,3	27,3	27,3	27,3	27,3	27,31	27,3	27,3	27,3
15	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3
16	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3
17	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3
18	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3
19	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3
20	27,3	27,3	27,31	27,3	27,31	27,3	27,31	27,3	27,3	27,3

Figura 20. MAPE E4

```
***************
OWA Brute Force Table

Tiempo de proceso: 2.501 segundos.

Brute-force     Right-shoulder  9,8532  ,1265 ,5010     [ ,9973 ,0027 ]
Brute-force     Linear  9,8532  ,0263 ,9948     [ ,9973 ,0027 ]
Brute-force     Power   9,8532  ,0038   [ ,9974 ,0026 ]

Tiempo de proceso BruteForce (0.0001): 109.683 segundos.

Brute-force     Right-shoulder  9,8532  ,1260 ,5010     [ ,9973 ,0027 ]
Brute-force     Linear  9,8532  ,2510 ,9960     [ ,9973 ,0027 ]
Brute-force     Power   9,8533  ,0040   [ ,9972 ,0028 ]

Tiempo de proceso BruteForce (0.001): 0.983 segundos.

Monte Carlo     Right-shoulder  9,8532  ,1476 ,5009     [ ,9973 ,0027 ]
Monte Carlo     Linear  9,8532  ,2826 ,9962     [ ,9973 ,0027 ]
Monte Carlo     Power   9,8532  ,0771   [ ,9479 ,0521 ]

Tiempo de proceso Montecarlo (500000): 1.716 segundos.
```

Figura 21. Resultados E5

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	9,85	9,85	9,85	9,85	9,85	9,85	10,4	11,2	12,2	13,1	14
0,1		9,85	9,85	9,85	9,85	9,85	10,6	11,7	12,9	14	15
0,2			9,85	9,85	9,85	9,85	10,9	12,5	14	15,4	17
0,3				9,85	9,85	9,85	11,7	14	16	17,4	19
0,4					9,85	9,85	14	17,4	19,4	20,7	22
0,5						9,85	26,8	26,8	26,8	26,8	27
0,6							26,8	26,8	26,8	26,8	27
0,7								26,8	26,8	26,8	27
0,8									26,8	26,8	27
0,9										26,8	27
1											27

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	14	13,2	12,5	11,9	11,3	10,9	10,6	10,3	10,1	9,9	9,9
0,1	15,1	14	13,1	12,4	11,7	11,1	10,7	10,4	10,1	9,91	9,9
0,2	16,5	15,2	14	13	12,2	11,5	10,9	10,5	10,2	9,93	9,9
0,3	18,5	16,9	15,4	14	12,9	12	11,2	10,7	10,3	9,95	9,9
0,4	21,6	19,4	17,4	15,7	14	12,7	11,7	10,9	10,4	9,99	9,9
0,5	26,8	23,5	20,7	18,2	16	14	12,5	11,3	10,6	10,1	9,9
0,6	26,8	24	21,6	19,4	17,4	15,7	14	12,7	11,7	10,9	10
0,7	26,8	24,4	22,3	20,3	18,5	16,9	15,4	14	12,9	12	11
0,8	26,8	24,7	22,8	21	19,4	17,9	16,5	15,2	14	13	12
0,9	26,8	24,9	23,2	21,6	20,1	18,7	17,4	16,2	15,1	14	13
1	26,8	25,1	23,5	22,1	20,7	19,4	18,2	17,1	16	15	14

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	9,94	10,2	10,5	10,9	11,3	11,8	12,3	12,9	13,4
1	14	14,7	15,3	15,9	16,4	17	17,5	18	18,5	19
2	19,4	19,8	20,2	20,6	21	21,3	21,6	21,9	22,2	22,5
3	22,8	23	23,3	23,5	23,7	23,9	24,1	24,2	24,4	24,5
4	24,7	24,8	24,9	25,1	25,2	25,3	25,4	25,5	25,6	25,6
5	25,7	25,8	25,8	25,9	26	26	26,1	26,1	26,2	26,2
6	26,2	26,3	26,3	26,3	26,4	26,4	26,4	26,4	26,5	26,5
7	26,5	26,5	26,5	26,5	26,6	26,6	26,6	26,6	26,6	26,6
8	26,6	26,6	26,6	26,7	26,7	26,7	26,7	26,7	26,7	26,7
9	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7
10	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7	26,7
11	26,7	26,7	26,7	26,7	26,7	26,7	26,8	26,8	26,8	26,8
12	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
13	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
14	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
15	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
16	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
17	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
18	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
19	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8
20	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8	26,8

Figura 22. MAPE E5

```
***************
OWA Brute Force Table

Tiempo de proceso: 2.586 segundos.

Brute-force     Right-shoulder  16,5820 ,5001 ,5001     [ ,0000 1,0000 ]
Brute-force     Linear  16,5820 ,5001 ,0000     [ ,0000 1,0000 ]
Brute-force     Power   16,5820 20,0000 [ ,0000 1,0000 ]

Tiempo de proceso BruteForce (0.0001): 101.943 segundos.

Brute-force     Right-shoulder  16,5820 ,5000 ,5000     [ ,0000 1,0000 ]
Brute-force     Linear  16,5820 ,5000 ,0000     [ ,0000 1,0000 ]
Brute-force     Power   16,5820 19,9990 [ ,0000 1,0000 ]

Tiempo de proceso BruteForce (0.001): 1.015 segundos.

Monte Carlo     Right-shoulder  16,5820 ,8772 ,9215     [ ,0000 1,0000 ]
Monte Carlo     Linear  16,5820 ,7235 ,0000     [ ,0000 1,0000 ]
Monte Carlo     Power   19,0585 20,0000 [ ,0000 1,0000 ]

Tiempo de proceso Montecarlo (500000): 1.169 segundos.
```

Figura 23. Resultados E6

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	22,2	22	22	22,2	22,2	22,2	21,2	20,4	19,8	19,39	19
0,1		22	22	22,2	22,2	22,2	20,9	20,1	19,5	19,06	19
0,2			22	22,2	22,2	22,2	20,6	19,7	19,1	18,63	18
0,3				22,2	22,2	22,2	20,1	19,1	18,5	18,06	18
0,4					22,2	22,2	19,1	18,1	17,6	17,37	17
0,5						22,2	16,6	16,6	16,6	16,58	17
0,6							16,6	16,6	16,6	16,58	17
0,7								16,6	16,6	16,58	17
0,8									16,6	16,58	17
0,9										16,58	17
1											17

q1\q2	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	19,1	19	20	20	20,3	20,6	20,9	21,3	21,6	21,88	22
0,1	18,7	19	19	19,7	20,1	20,5	20,8	21,2	21,5	21,84	22
0,2	18,3	19	19	19,4	19,8	20,2	20,6	21	21,4	21,8	22
0,3	17,8	18	19	19,1	19,5	19,9	20,4	20,9	21,3	21,75	22
0,4	17,2	18	18	18,6	19,1	19,6	20,1	20,6	21,2	21,67	22
0,5	16,6	17	17	17,9	18,5	19,1	19,7	20,3	20,9	21,57	22
0,6	16,6	17	17	17,6	18,1	18,6	19,1	19,6	20,1	20,63	21
0,7	16,6	17	17	17,4	17,8	18,2	18,6	19,1	19,5	19,94	20
0,8	16,6	17	17	17,3	17,6	17,9	18,3	18,7	19,1	19,43	20
0,9	16,6	17	17	17,2	17,5	17,7	18,1	18,4	18,7	19,06	19
1	16,6	17	17	17,2	17,4	17,6	17,9	18,2	18,5	18,76	19

q	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0	?	22	21	21	20,7	20,4	20	19,8	19,5	19,27
1	19,1	19	19	18,5	18,3	18,2	18	17,9	17,8	17,7
2	17,6	18	17	17,4	17,3	17,3	17,2	17,2	17,1	17,09
3	17	17	17	17	16,9	16,9	16,9	16,9	16,8	16,83
4	16,8	17	17	16,8	16,8	16,7	16,7	16,7	16,7	16,7
5	16,7	17	17	16,7	16,7	16,7	16,7	16,7	16,6	16,64
6	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,61
7	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,6
8	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,59
9	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,59
10	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
11	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
12	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
13	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
14	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
15	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
16	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
17	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
18	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
19	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58
20	16,6	17	17	16,6	16,6	16,6	16,6	16,6	16,6	16,58

Figura 24. MAPE E6

4.2.5 Comparativa resultados

En la Tabla 1, para cada experimento, algoritmo y tipo de cuantificador, se muestra el mejor MAPE, los mejores parámetros y el vector de ponderación correspondiente. Para cada experimento y algoritmo, se muestra el tiempo total de ejecución (en segundos) para optimizar los parámetros de todos los tipos de cuantificadores.

Experiment	Algorithm	Quantifier	MAPE	Parameters	Weights	Time
E1	Brute-force $\Delta = 0.0001$	Right-shoulder	**5.3888**	0.1991 0.3851	[0.0048 0.9952 0 0 0]	47
		Linear	9.4604	0.4434 0.9319	[0.4203 0.4203 0.1104 0.0245 0.0245]	
		Power	10.0735	0.3014	[0.6156 0.143 0.0986 0.0777 0.065]	
	Brute-force $\Delta = 0.001$	Right-shoulder	5.3956	0.199 0.4	[0.005 0.995 0 0 0]	0.6
		Linear	9.4614	0.443 0.932	[0.4208 0.4208 0.1096 0.0244 0.0244]	
		Power	10.0741	0.301	[0.616 0.1429 0.0985 0.0776 0.065]	
	Monte Carlo (500000)	Right-shoulder	5.3893	0.1995 0.3007	[0.0048 0.9952 0 0 0]	0.7
		Linear	9.4613	0.4421 0.9309	[0.4212 0.4212 0.1081 0.0248 0.0248]	
		Power	10.0735	6.0285	[0.0001 0.0039 0.0420 0.2145 0.7395]	
E2	Brute-force $\Delta = 0.0001$	Right-shoulder	**5.2896**	0.166 0.513	[0.0019 0.4803 0.4803 0.0375 0 0]	52
		Linear	12.6139	0.0836 0.0082	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	13.4133	102.431	[0.1078 0.1474 0.1673 0.1816 0.1931 0.2028]	
	Brute-force $\Delta = 0.001$	Right-shoulder	**5.2896**	0.166 0.513	[0.0019 0.4803 0.4803 0.0375 0 0]	0.5
		Linear	12.6139	0.125 0.053	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	13.4136	10.243	[0.1078 0.1474 0.1673 0.1816 0.1931 0.2028]	
	Monte Carlo (500000)	Right-shoulder	5.29	0.166 0.5129	[0.0019 0.4805 0.4805 0.0372 0 0]	0.6
		Linear	12.6139	0.1447 0.0743	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	16.0601	19.9999	[0 0 0 0.0003 0.0258 0.9739]	
E3	Brute-force $\Delta = 0.0001$	Right-shoulder	**8.7904**	0.5605 0.8303	[0 0 0 0.3935 0.6065 0]	228
		Linear	9.5211	0.0836 0.0082	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	9.7050	1.5976	[0.0571 0.1158 0.1575 0.1928 0.2241 0.2527]	
	Brute-force $\Delta = 0.001$	Right-shoulder	**8.7904**	0.166 0.513	[0 0 0 0.3935 0.6065 0]	2.2
		Linear	9.5216	0.125 0.053	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	9.7050	1.5980	[0.0571 0.1157 0.1575 0.1928 0.2241 0.2527]	
	Monte Carlo (500000)	Right-shoulder	**8.7904**	0.166 0.5129	[0 0 0 0.3935 0.6065 0]	2.3
		Linear	9.5220	0.1447 0.0743	[0.0981 0.1804 0.1804 0.1804 0.1804 0.1804]	
		Power	10.8831	20	[0 0 0 0.003 0.0258 0.9739]	
E4	Brute-force $\Delta = 0.0001$	Right-shoulder	9.007	0.1492 0.5278	[0.9266 0.0734]	102
		Linear	9.007	0.0535 0.861	[0.9266 0.0734]	
		Power	9.007	0.11	[0.9266 0.0734]	
	Brute-force $\Delta = 0.001$	Right-shoulder	9.007	0.235 0.521	[0.9266 0.0734]	1
		Linear	9.007	0.142 0.874	[0.9266 0.0734]	
		Power	9.007	0.11	[0.9266 0.0734]	
	Monte Carlo (500000)	Right-shoulder	9.007	0.073 0.5338	[0.9266 0.0734]	1.1
		Linear	9.007	0.059 0.8618	[0.9266 0.0734]	
		Power	9.007	2.2004	[0.2176 0.0734]	
E5	Brute-force $\Delta = 0.0001$	Right-shoulder	9.8532	0.1265 0.501	[0.9973 0.0027]	110
		Linear	9.8532	0.0263 0.9948	[0.9973 0.0027]	
		Power	9.8532	0.0038	[0.9974 0.0026]	
	Brute-force $\Delta = 0.001$	Right-shoulder	9.8532	0.1476 0.5009	[0.9973 0.0027]	1
		Linear	9.8532	0.2826 0.9962	[0.9973 0.0027]	
		Power	9.8533	0.004	[0.9972 0.0028]	
	Monte Carlo (500000)	Right-shoulder	9.8532	0.1476 0.5009	[0.9973 0.0027]	1.7
		Linear	9.8532	0.2826 0.9962	[0.9973 0.0027]	
		Power	9.8532	0.0771	[0.9973 0.0027]	
E6	Brute-force $\Delta = 0.0001$	Right-shoulder	16.582	0.5 0.5	[0 1]	102
		Linear	16.582	0.5 0	[0 1]	
		Power	16.582	20	[0 1]	
	Brute-force $\Delta = 0.001$	Right-shoulder	16.582	0.5 0.5	[0 1]	1
		Linear	16.582	0.5 0	[0 1]	
		Power	16.582	20	[0 1]	
	Monte Carlo (500000)	Right-shoulder	16.582	0.6822 0.842	[0 1]	1.1
		Linear	16.582	0.7235 0	[0 1]	
		Power	16.582	20	[0 1]	

Tabla 1. Resultado de los experimentos

Para empezar, conviene aclarar que el principal objetivo no es resolver algunos problemas de predicción sino encontrar los parámetros correspondientes a las mejores ponderaciones OWA. De hecho, para resolver estos problemas de predicción, es muy probable que las estrategias de aprendizaje automático más complejas funcionen con mejores resultados.

La primera observación interesante es que el MAPE es muy similar independientemente del algoritmo para un tipo de cuantificador dado. Es más, en todos los experimentos con el conjunto de datos de Turismo (E4, E5 y E6), el MAPE es en realidad el mismo (y también lo es el vector de ponderación) independientemente del tipo de cuantificador. En los otros experimentos (E1, E2 y E3), las diferencias en el MAPE son menores que 0.007.

A modo de comparación, se destaca:

- El método Fuerza bruta con $\Delta = 0{,}0001$ siempre tiene el MAPE más pequeño. Fuerza bruta con un incremento mayor es ligeramente peor en el caso del E1 (para el resto, es igual).
- Monte Carlo es ligeramente peor en E1 y E2 (el mismo para el resto), pero los MAPE son realmente iguales si se redondea a dos decimales.

Los tiempos de ejecución, sin embargo, son más diferentes entre sí. Los algoritmos más rápidos son de Fuerza bruta con $\Delta = 0{,}001$ y Montecarlo. Ambos con casi el mismo tiempo. Por otro lado, Fuerza bruta con $\Delta = 0{,}0001$ es bastante más lento, observándose que dicho aumento en el tiempo de ejecución no se compensa con una disminución significativa en el MAPE.

Si se comparan los tipos de cuantificadores, Right-Shoulder es siempre la mejor función en el conjunto de datos de la aplicación 1, mientras que en el conjunto de datos de las aplicaciones 2 y 3, siempre se obtiene el mismo MAPE independientemente del tipo de cuantificador.

En relación a la comparativa de cada una de las aplicaciones por separado, se destaca:

- Primera aplicación (E1, E2).

El MAPE es, en general, claramente más pequeño en E1 que en E2. Este resultado es el esperado, porque E2 introduce datos (de Buenos Aires) que dificultan la predicción. Sin embargo, los experimentos con la función Right-Shoulder son una excepción, y el MAPE es en realidad un poco más pequeño en E2. Se puede notar que, en estos casos, el peso asociado al menor valor a agregar es 0, y el peso asociado al valor más grande es muy pequeño, aproximadamente 0,002. Debido a que Buenos Aires es la única ciudad del hemisferio sur, su temperatura generalmente será la más alta o la más baja, pero tendrá una pequeña influencia en el valor agregado debido a los pesos. Se ha de tener en cuenta que, después de considerar Buenos Aires, el tamaño del vector de ponderación aumenta. Este es el motivo por el que, para construir los vectores de ponderación, se evalúan (Ecuación 1) las funciones cuantificadoras en los valores {0, 0.2, 0.4, 0.6, 0.8, 1} en lugar de en los valores {0, 0.25, 0.5, 0.75, 1}.

- Segunda aplicación (E3).

Para los tres tipos de funciones (Fuerza bruta $\Delta = 0{,}001$, Fuerza bruta $\Delta = 0{,}0001$ y Montecarlo), el resultado del MAPE, así como el vector de pesos asociados a cada una de las entradas, es el mismo (excepto una excepción en la función Power en Montecarlo). La única diferencia radica en los valores de q_1 y q_2.

- Tercera aplicación (E3, E4, E5).

La mejor estrategia parece utilizar casi exclusivamente una de las dos variables de entrada. De hecho, en E6, el vector de ponderación es [0,1]. En los otros experimentos E4 y E5, el valor más alto a agregar tiene un peso mayor que 0.9. El MAPE es peor que en el conjunto de datos de las aplicaciones primera y segunda, y es particularmente alto cuando se predice el gasto total (E6).

5. Conclusiones y trabajo futuro

Una vez finalizada la elaboración del proyecto, es el momento de evaluar algunos aspectos como el alcance de los resultados, las dificultades afrontadas, el aprendizaje adquirido... Además, de cara a un futuro, cabe comentar algunas ideas que sería interesante introducir para la mejora del proyecto.

5.1 Conclusiones

En este proyecto, el método empleado puede considerarse como un híbrido de dos categorías de algoritmos, los basados en funciones (cuantificadores RIM) y los basados en datos empíricos. Esto lo hace todavía más interesante. Por ello, se ha seguido un enfoque híbrido para la adjudicación de los pesos de los operadores OWA, eligiendo los parámetros de algunas funciones, comúnmente utilizadas en la agregación guiada por cuantificadores.

Se han estudiado las funciones Right-Shoulder, Linear y Power, y propuesto dos alternativas para la búsqueda en el espacio de parámetros: Fuerza bruta y un algoritmo de Monte Carlo. Siempre con el objetivo de observar el conjunto de parámetros y funciones que hacen minimizar el error acumulado, para cada una de las aplicaciones.

El enfoque de este TFG tiene varias ventajas sobre el aprendizaje directo de los pesos OWA. Por un lado, el número de parámetros es menor, reduciendo el espacio de búsqueda. Por otro lado, los resultados son más interpretables, ya que los cuantificadores difusos pueden ser más fácilmente entendibles por el ser humano.

También se han discutido los resultados de una evaluación empírica sobre tres conjuntos de datos (tres aplicaciones) en el campo de las ciudades inteligentes. Se encuentran diferencias significativas en los tiempos de ejecución de los algoritmos, pero no en el error producido. Además, se observa que, en dos conjuntos de datos, el error es menor para la función Right-Shoulder, mientras que en el otro conjunto de datos los resultados fueron independientes del tipo de función.

Cabe resaltar que este trabajo ha sido aceptado, como artículo, para su presentación en el congreso como *"Learning OWA weights by combining fuzzy quantifiers with empirical data*" (ESTYLF 20-21), y será incluido en el correspondiente libro de actas.

5.2 Posibles alternativas y trabajo futuro

En el proyecto, hay muchas direcciones para el trabajo futuro.

En primer lugar, sería interesante considerar más tipos de RIMS.

En segundo lugar, se podría considerar algoritmos más complejos para buscar los mejores parámetros. También se podría tener en cuenta enfoques alternativos a la agregación guiada por cuantificadores (véase sección 2.4).

Por último, sería deseable realizar experimentos con conjuntos de datos más reales, provenientes de fuentes lo mayor fiables posibles.

Referencias bibliográficas

[1] Álvaro Cristóbal, Ignacio Huitzil, Fernando Bobillo, Submitted, https://easychair.org/conferences/?conf=estylf20-21 , 2021.

[2] Schema Organization, "Ciudades inteligentes", https://www.esmartcity.es/ciudades-inteligentes , 9 enero 2018.

[3] Mario Vaquero, "¿Qué son las Smart Cities o ciudades inteligentes?", https://panelesach.com/blog/smart-cities-o-ciudades-inteligentes-que-son/ , 9 enero 2018.

[4] Organization for Economic Co-operation and Development, "*OECD Environmental Outlook*", https://www.oecd.org/environment/indicators-modelling-outlooks/oecd-environmental-outlook-1999155x.htm , 5 abril 2001.

[5] Pascual Berrone, Joan Enric Ricart (*IESE Cities in Motion*), https://media.iese.edu/research/pdfs/ST-0509.pdf , 2019.

[6] Ana Torres Pastor, "Las Smart Cities y su implementación, los casos de España y Dinamarca", https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/23754/TFG%20Final%20Ana%20Torres.pdf?sequence=1&isAllowed=y , junio 2018.

[7] Acciona Organisation, "¿Qué es una Smart City? Top 5 ciudades inteligentes", https://www.sostenibilidad.com/construccion-y-urbanismo/que-es-una-smart-city-top-5-ciudades-inteligentes/ , 2019.

---

[8] V. Torra and Y. Narukawa. *Modeling decisions - Information fusion and aggregation operators*. Springer, 2007.

[9] G. Beliakov, H. Bustince, and T. Calco, "*A practical guide to averaging functions", ser. Studies in Fuzziness and Soft Computing*. Springer, 2016, vol. 329.

[10] X. Liu, "A review of the OWA determination methods: Classification and some extensions"*, in Recent Developments in the Ordered Weighted Averaging Operators: Theory and Practice, ser. Studies in Fuzziness and Soft Computing.* Springer, 2011, vol. 265, pp. 49–90.

[11] Z. Xu, "*An overview of methods for determining OWA weights*", *International Journal of Intelligent Systems*, vol. 20, no. 8, pp. 843–865, 2005.

[12] R. Fullér, *"On obtaining OWA operator weights: A short survey of recent developments"*, *in Proceedings of the 5th International Conference on Computational Cybernetics (ICCC 2007)*, 2007, pp. 241–244.

[13] R. R. Yager, "*On ordered weighted averaging aggregation operators in multicriteria decision making*", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

[14] R. R. Yager, J. Kacprzyk, and G. Beliakov, Eds., "*Recent developments in the Ordered Weighted Averaging operators: Theory and practice*", *ser. Studies in Fuzziness and Soft Computing.* Springer, 2011, vol. 265.

[15] R. R. Yager, "*Quantifier guided aggregation using OWA operators*", *International Journal of Intelligent Systems*, vol. 11, no. 1, pp. 49–73, 1996.

[16] F. Bobillo and U. Straccia, "*Fuzzy ontology representation using OWL 2", International Journal of Approximate Reasoning*, vol. 52, no. 7, pp. 1073–1094, 2011.

[17] F. Bobillo and U. Straccia, "*The fuzzy ontology reasoner fuzzyDL*", *Knowledge-Based Systems*, vol. 95, pp. 12–34, 2016.

[18] I. Huitzil, F. Alegre, and F. Bobillo, "GimmeHop: "*A recommender system for mobile devices using ontology reasoners and fuzzy logic*", *Fuzzy Sets and Systems*, vol. 401, pp. 55–77, 2020.

[19] I. Huitzil, M. Molina-Solana, J. Gómez-Romero, and F. Bobillo, "*Minimalistic fuzzy ontology reasoning: An application to Building Information Modeling*", *Applied Soft Computing*, vol. 103, p. 107158, 2021.

[20] R. R. Yager, "*Connectives and quantifiers in fuzzy sets*", *Fuzzy Sets and Systems*, vol. 40, no. 1, pp. 39–75, 1991.

[21] Matías Lorenzón, David L. de la Red Martínez, "Operadores de Agregación", http://exa.unne.edu.ar/informatica/SO/Lorenzon_Matias-Operadores_de_Agregacion.pdf pp. 25, 23 marzo 2011.

[22] Recursos En Project Management Organization, "Método Montecarlo en proyectos", https://www.recursosenprojectmanagement.com/metodo-de-montecarlo/#%C2%BFQue_es_el_metodo_de_Montecarlo , 17 enero 2015.

[23] José Ignacio Illana, "Métodos Monte Carlo", https://www.ugr.es/~jillana/Docencia/FM/mc.pdf , enero 2013.

[24] Krishnan Saravanan, E. Golden Julie and Y. Harold Robinson, "*Smart Cities & IoT: Evolution of Applications, Architectures & Technologies, Present Scenarios & Future Dream*", chapter 7.

[25] L. Troiano and R. R. Yager, "Recursive and iterative OWA operators, *"International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems"*, vol. 13, no. 6, pp. 579–600, 2005.

[26] O. Duarte and S. Téllez, "*A family of OWA operators based on Faulhaber's formulas"*, arXiv 1801.10545, 2018.

---

[27] Wikipedia, "Mean absolute percentage error", https://en.wikipedia.org/wiki/Mean_absolute_percentage_error , diciembre 2009.

[28] Gestión de Operaciones Tutoriales Organización, "Error Porcentual Absoluto Medio (MAPE) en un Pronóstico de Demanda", https://www.gestiondeoperaciones.net/proyeccion-de-demanda/error-porcentual-absoluto-medio-mape-en-un-pronostico-de-demanda/ , 26 enero 2015.

[29] Ingenio Empresa Organización, "Medición del error de pronósticos de demanda", https://www.ingenioempresa.com/medicion-error-pronostico/#:~:text=Error%20porcentual%20medio%20absoluto%20(MAPE),-El%20MAPE%20nos&text=Es%20el%20promedio%20del%20error,porcentaje%20de%20los%20valores%20reales , 7 marzo 2016.

[30] Mohamad, "MAPE - Error medio de porcentaje absoluto", https://support.numxl.com/hc/es/articles/215959443-MAPE-Error-medio-de-porcentaje-absoluto , 26 octubre 2016.

---

[31] Ayuntamiento de Madrid, "Accidentes de tráfico. Datos desde 2009 a 2016 (Seguridad vial)", https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=746b86396f847410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default , 21 julio 2014.

[32] *New York Open Data*, "*Air Quality*", https://data.cityofnewyork.us/Environment/Air-Quality/c3uy-2p5r , 24 octubre

2014.

[33] *New York Open Data*, “Infant Mortality”, https://data.cityofnewyork.us/Health/Infant-Mortality/fcau-jc6k , 19 abril 2017.

[34] *Helsinki Region Infoshare*, “*Noise sensor data from Helsinki*”, https://hri.fi/data/en_GB/dataset/iot-meludataa-helsingista, 27 mayo 2020.

---

[35] Gobierno de Buenos Aires, “Registro de Temperatura”, https://data.buenosaires.gob.ar/dataset/registro-temperatura-ciudad , 10 mayo 2021.

[36] Comune di Milano, ”Meteo: temperatura per mese”, http://dati.comune.milano.it/dataset/ds305-ambientemeteo-temperature-mese-2008-2014 , 27 noviembre 2015.

[37] Ajuntament de Barcelona (Servei Meteorològic de Catalunya), “Temperaturas medianas mensuales del aire de la ciudad de Barcelona desde 1780”, http://opendata-ajuntament.barcelona.cat/data/es/dataset/temperatures-hist-bcn , 22 octubre 2019.

[38] Climate Data Organization, “Madrid: Tiempo y clima en Enero”. http://es.climate-data.org/europe/espana/comunidad-de-madrid/madrid-92/t/enero-1 , (10 junio 2021).

[39] Meteoblue Organization, “Acceso demo de Basilea”, https://www.meteoblue.com/es/tiempo/archive/export/basilea_suiza_2661604 , (10 junio 2021).

[40] Gobierno de La Rioja, “Estaciones meteorológicas SOS Rioja”, https://datos.gob.es/en/catalogo/a17002943-estaciones-meteorologicas-sos-rioja1 , 9 octubre 2015.

[41] AEMET, “Estaciones Meteorológicas de España (Datos desde Mayo-2013)“, http://datosclima.es/Aemet2013/Temperatura2013.php , (10 junio 2021).

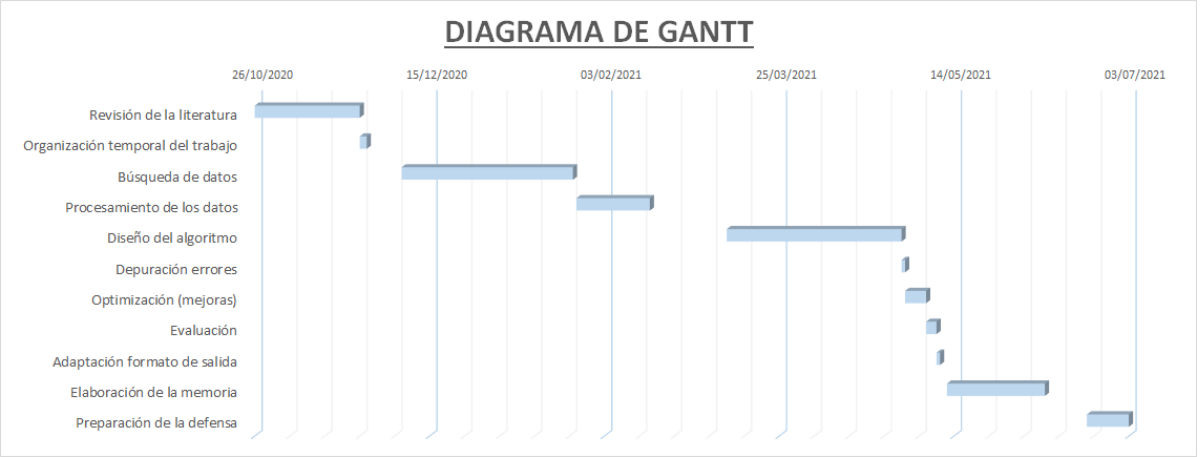
[42] AEMET, , https://datosclima.es , (10 junio 2021).

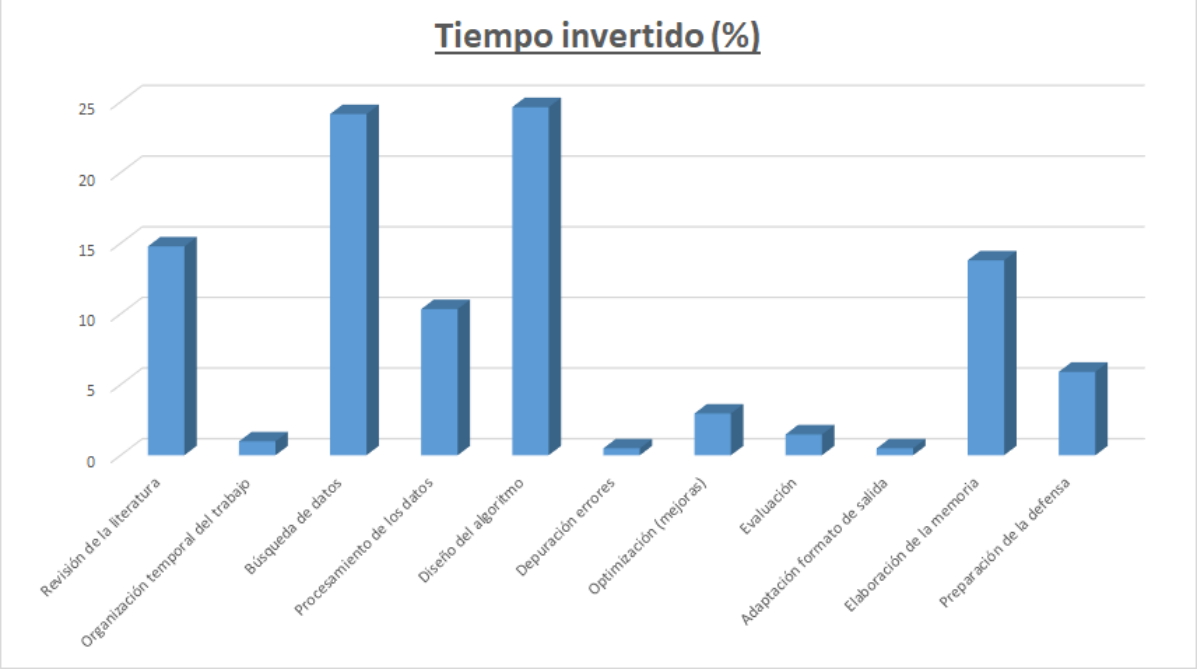
[43] Office for National Statistics (ONS), London, “*Number of International Visitors to London*”, https://data.london.gov.uk/dataset/number-international-visitors-london , 1 enero 2002.

6. Anexos

6.1 Tiempo empleado

A continuación, se establecen los plazos temporales empleados para cada una de las tareas de las que ha constado el proyecto, así como el porcentaje de número de horas invertidas en cada una de ellas.





6.2 Código fuente

Se proporciona el código fuente empleado, para la realización del proyecto:

```
1
2 import java.io.*;
3 import java.text.*;
4 import java.util.*;
5 import org.apache.poi.openxml4j.exceptions.*;
6 import org.apache.poi.ss.usermodel.*;
7 import org.apache.poi.xssf.usermodel.*;
8
9 public class TRABAJO_FINAL
10 {
11
12     ArrayList<Double> datosParaComparar;
13     private DecimalFormat df;
14     ArrayList<Double>[] filass;
15     private int numFilas;
16
17
18     public static double owa(Double[] w, ArrayList<Double> x)
19     {
20         Comparator<Double> comparador = Collections.reverseOrder(); // ordenar de mayor a menor para owa
21         Collections.sort(x, comparador);
22         double owa = 0;
23         for (int i = 0; i < x.size(); i++)
24             owa += w[i] * x.get(i);
25         return owa;
26     }
27
28
29     public static double weightedSum(Double[] w, ArrayList<Double> x)
30     {
31         double ws = 0;
32         for (int i = 0; i < x.size(); i++)
33             ws += w[i] * x.get(i);
34         return ws;
35     }
36
37      //Cálculo de los pesos (usado en todos los siguientes cuantificadores)
38      public static Double[] getWvector(int numWeights, double[] fVector)
39      {
40          Double[] w = new Double[numWeights];
41          for (int i = 0; i < numWeights; i++)
42              w[i] = fVector[i + 1] - fVector[i];
43          return w;
44      }
45
46
```

```java
    //MÉTODO 1 (Right-Shoulder):
    public static Double[] rightShoulderQuantifier(int numWeights, double q1, double q2)
    {
        double[] fVector = new double[numWeights + 1];
        double n = 1 / (double) numWeights; //Normalizado
        double in = 0;
        double aux = 0;

        if (q1 == q2)
        {
            fVector[0] = 0;
            for (int i = 1; i < numWeights; i++)
            {
                in = in + n; //Monótona creciente
                if (in < q1)
                    fVector[i] = 0;
                else
                    fVector[i] = 1;
            }
            fVector[numWeights] = 1;
        }
        else
        {
            for (int i = 0; i < numWeights + 1; i++)
            {
                if (i != 0)
                    in = in + n; //Monótona creciente

                aux = (in - q1) / (q2 - q1);

                //Evaluamos los tres posibles rangos
                if (aux < 0)
                    fVector[i] = 0;
                else if (aux > 1)
                    fVector[i] = 1;
                else
                    fVector[i] = aux;
            }
        }

        return getWvector(numWeights, fVector);
    }

```

```java
//MÉTODO 2 (Linear):
public static Double[] linearQuantifier(int numWeights, double q1, double q2)
{
    double[] fVector = new double[numWeights + 1];
    double n = 1 / (double) numWeights; //Normalizado
    double in = 0;

    if (q1 == 0)
    {
        fVector[0] = 0;
        for (int i = 1; i < numWeights + 1; i++)
        {
            in = in + n;
            fVector[i] = (in - 1) * (1 - q2) + 1;
        }
    }
    else if (q1 == 1)
    {
        for (int i = 0; i < numWeights; i++)
        {
            if (i != 0)
                in = in + n;
            fVector[i] = (in * q2);
        }
        fVector[numWeights] = 1;
    }
    else
    {
        for (int i = 0; i < numWeights; i++)
        {
            if (i != 0)
                in = in + n;
            if (in < q1)
                fVector[i] = (in * q2) / q1;
            else
                fVector[i] = (in - 1) * (1 - q2) / (1 - q1) + 1;
        }
        fVector[numWeights] = 1;
    }
    return getWvector(numWeights, fVector);
}
```

```java
    //MÉTODO 3 (Power)
    public static Double[] powerQuantifier(int numWeights, double q)
    {
        double[] fVector = new double[numWeights + 1];
        double n = 1 / (double) numWeights; //Normalizado
        double in = 0;
        for (int i = 0; i < numWeights + 1; i++)
        {
            if (i != 0)
                in = in + n;
            fVector[i] = Math.pow(in, q);
        }
        return getWvector(numWeights, fVector);
    }


    //-----------------------------------------------------------------------------------

    public static double calculoOwaRS(ArrayList<Double> valores,Double[] rightShoulder,double q1,double q2) {

        //Double[] rightShoulder = rightShoulderQuantifier(valores.size(), q1, q2);
        return owa(rightShoulder, valores);
    }


    public static double calculoOwaLinear(ArrayList<Double> valores,Double[] linear,double q1,double q2) {

        //Double[] linear = linearQuantifier(valores.size(), q1, q2);
        return owa(linear,valores);
    }


    public static double calculoOwaPower(ArrayList<Double> valores,Double[] power,double q) {

        //Double[] power = powerQuantifier(valores.size(), q);
        return owa(power,valores);
    }


    //-----------------------------------------------------------------------------------


    public void printVector (Double[] v)
    {
        System.out.print("[ ");
        for (int i=0; i<v.length; i++)
            System.out.print(String.valueOf(df.format(v[i])) + " ");
        System.out.println("]");
    }


    //-----------------------------------------------------------------------------------
```

```java
public TRABAJO_FINAL (File fileName, int numColumnas, int numFilas, int columnaInicial)
{

    this.numFilas = numFilas;
    try
    {
        FileInputStream entrada = new FileInputStream(fileName);
        XSSFWorkbook workbook = new XSSFWorkbook(entrada);
        df = new DecimalFormat ("#.0000");

        //Datos de entrada
        ArrayList<ArrayList<Double>> listaEntradas = new ArrayList<ArrayList<Double>>();

        for (int i = columnaInicial; i < (columnaInicial + numColumnas); i++)
        {
            listaEntradas.add(lecturaDatosExcel(workbook, entrada, i));
        }

        filass = new ArrayList [numFilas];

        for (int i = 0; i < numFilas; i++)
        {
            ArrayList<Double> filas = new ArrayList ();
            for (int j = 0; j < numColumnas; j++)
            {
                filas.add(listaEntradas.get(j).get(i));
            }
            filass[i] = filas;
        }

        //Datos de salida
        datosParaComparar = lecturaDatosExcel(workbook, entrada, columnaInicial + numColumnas);

        entrada.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

```java
  try
  {

      double incremento = 0.1; //Modificar según se quiera

      for (double q1 = 0; q1 <= 1; q1 += incremento)
      {
          for (double q2 = 0; q2 <= 1; q2 += incremento)
          {

              FileInputStream entrada = new FileInputStream(fileName);

              BufferedWriter salida_txt = new BufferedWriter(new FileWriter(new File("Ficheros salida TXT/SalidaResultados "
                      + "(q1=" +Math.round(100.0*q1)/100.0+ ", q2=" +Math.round(100.0*q2)/100.0+ ") .txt")));

              FileOutputStream salida_xlsx = new FileOutputStream ("Ficheros salida XLSX/SalidaResultados "
                      + "(q1=" +Math.round(100.0*q1)/100.0+ ",  q2=" +Math.round(100.0*q2)/100.0+ ") .xlsx");

              //obtenerDatosExcel(fileName);

              XSSFWorkbook workbook = new XSSFWorkbook(entrada);

              escrituraResultadosTXT (workbook, entrada, salida_txt, numeroEntradas, numeroMuestras, columnaInicial, q1, q2);
              escrituraResultadosXLSX (workbook, entrada, salida_xlsx, numeroEntradas, numeroMuestras, columnaInicial, q1, q2);

              entrada.close();

              salida_txt.close();
              salida_xlsx.close();

              //salida_comparativa_txt.close();
              //salida_comparativa_xlsx.close();

          }
      }
  }
  catch(Exception e)
  {
      e.printStackTrace();
  }
```

```java
public void owaBruteForceTabla (double incremento)
{
    System.out.println();
    System.out.println("***************\nOWA Brute Force Table\n");

    try
    {
        BufferedWriter salida = new BufferedWriter(new FileWriter(new File("RESULTADOS FINALES/FinalBrute_TXT.txt")));
        DecimalFormat df1 = new DecimalFormat ("#.0");


        //Primera representación (RIGHT-SHOULDER)

        salida.write("q1\\q2\t");
        for (double q2 = 0; q2 <= 1; q2 += incremento)
            salida.write(puntosPorComas(String.valueOf(df1.format(q2))) + "\t");
        salida.write("\n");

        for (double q1 = 0; q1 <= 1; q1 += incremento)
        {
            salida.write(puntosPorComas(String.valueOf(df1.format(q1))) + "\t");
            for (double q2 = 0; q2 <= 1; q2 += incremento)
            {
                Double[] rightShoulder = rightShoulderQuantifier(filass[0].size(), q1, q2);

                if (q1 > q2)
                    salida.write("\t");
                else
                {
                    double errorAcumuladoRS = 0.0;
                    for (int i = 0; i < numFilas; i++)
                    {
                        double resultadoRS = calculoOwaRS(filass[i], rightShoulder, q1, q2);
                        errorAcumuladoRS += Math.abs((resultadoRS - datosParaComparar.get(i)) / resultadoRS);
                    }
                    double errorMapeRS = (errorAcumuladoRS / numFilas) * 100;
                    salida.write(puntosPorComas(String.valueOf(df.format(errorMapeRS))) + "\t");
                }
            }
            salida.write("\n");
        }
        salida.write("\n");

        //Segunda representación (LINEAR)

        salida.write("q1\\q2\t");
        for (double q2 = 0; q2 <= 1; q2 += incremento)
            salida.write(puntosPorComas(String.valueOf(df1.format(q2))) + "\t");
        salida.write("\n");

        for (double q1 = 0; q1 <= 1; q1 += incremento)
        {
            salida.write(puntosPorComas(String.valueOf(df1.format(q1))) + "\t");
            for (double q2 = 0; q2 <= 1; q2 += incremento)
            {
                Double[] linear = linearQuantifier(filass[0].size(), q1, q2);
                double errorAcumuladoL = 0.0;
                for (int i = 0; i < numFilas; i++)
                {
                    double resultadoL = calculoOwaLinear(filass[i], linear, q1, q2);
                    errorAcumuladoL += Math.abs((resultadoL - datosParaComparar.get(i)) / resultadoL);
                }
                double errorMapeL = (errorAcumuladoL / numFilas) * 100;
                salida.write(puntosPorComas(String.valueOf(df.format(errorMapeL))) + "\t");
            }
            salida.write("\n");
        }
        salida.write("\n");
```

```
            //Tercera representación (POWER)

            salida.write("q\t"); // Entera\\Decimal
            for (int k = 0; k <= 9; k++)
                salida.write("," + k + "\t");

            int contador = 0;
            for (double q1 = 0; q1 < 21; q1 += incremento)
            {
                Double[] power = powerQuantifier(filass[0].size(), q1);

                if (contador % 10 == 0)
                    salida.write("\n" + (contador / 10) + "\t");

                double errorAcumuladoP = 0.0;
                for (int i = 0; i < numFilas; i++)
                {
                    double resultadoP = calculoOwaPower(filass[i], power, q1);
                    errorAcumuladoP += Math.abs((resultadoP - datosParaComparar.get(i)) / resultadoP);
                }
                double errorMapeP = (errorAcumuladoP / numFilas) * 100;
                salida.write(puntosPorComas(String.valueOf(df.format(errorMapeP))) + "\t");
                contador++;
            }
            salida.write("\n");

            salida.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }



    // ---------------- PRINTS ----------------


    // FORMA 1/3
    public void owaBruteForce (double incremento)
    {
        double bestErrorRS = 100;
        double bestErrorL = 100;
        double bestErrorP = 100;
        double bestQ1R = 0;
        double bestQ2R = 0;
        double bestQ1L = 0;
        double bestQ2L = 0;
        double bestQ = 0;
        int numCols = filass[0].size();

        // 1. Cálculo de los errores

        for (double q1 = 0; q1 < 1; q1 += incremento)
        {
            for (double q2 = 0; q2 < 1; q2 += incremento)
            {
                Double[] rightShoulder = rightShoulderQuantifier(filass[0].size(), q1, q2);
                Double[] linear = linearQuantifier(filass[0].size(), q1, q2);

                Double errorMapeRS = null;
                Double errorMapeL = null;
```

```
366                 if (q1 <= q2)
367                 {
368                     double errorAcumuladoRS = 0.0;
369                     for (int i = 0; i < numFilas; i++)
370                     {
371                         double resultadoRS = calculoOwaRS(filass[i], rightShoulder, q1, q2);
372                         errorAcumuladoRS += Math.abs((resultadoRS - datosParaComparar.get(i)) / resultadoRS);
373                     }
374                     errorMapeRS = (errorAcumuladoRS / numFilas) * 100;
375                     if (errorMapeRS < bestErrorRS)
376                     {
377                         bestErrorRS = errorMapeRS;
378                         bestQ1R = q1;
379                         bestQ2R = q2;
380                     }
381                 }
382 
383                     double errorAcumuladoL = 0.0;
384                     for (int i = 0; i < numFilas; i++)
385                     {
386                         double resultadoL = calculoOwaLinear(filass[i], linear, q1, q2);
387                         errorAcumuladoL += Math.abs((resultadoL - datosParaComparar.get(i)) / resultadoL);
388                     }
389                     errorMapeL = (errorAcumuladoL / numFilas) * 100;
390                     if (errorMapeL < bestErrorL)
391                     {
392                         bestErrorL = errorMapeL;
393                         bestQ1L = q1;
394                         bestQ2L = q2;
395                     }
396 //              }
397             }
398         }
399 
400         for (double q = 0; q < 20; q += incremento)
401         {
402             //System.out.println("q: " + q);
403 
404             Double[] power = powerQuantifier(filass[0].size(), q);
405             Double errorMapeP = null;
406 
407             //double resultadoP = calculoOwaPower(filass[0], q);
408             double errorAcumuladoP = 0.0;
409             for (int i = 0; i < numFilas; i++)
410             {
411                 double resultadoP = calculoOwaPower(filass[i], power, q);
412                 errorAcumuladoP += Math.abs((resultadoP - datosParaComparar.get(i)) / resultadoP);
413             }
414             errorMapeP = (errorAcumuladoP / numFilas) * 100;
415             if (errorMapeP < bestErrorP)
416             {
417                 bestErrorP = errorMapeP;
418                 bestQ = q;
419             }
420         }
421 
```

```
422         // 2. Escritura de los resultados
423 
424         System.out.println();
425 //      System.out.println("***************\nOWA Brute Force\n");
426         System.out.print("Brute-force\tRight-shoulder\t");
427         System.out.print(
428                 puntosPorComas(String.valueOf(df.format(bestErrorRS))) + "\t" +
429                 puntosPorComas(String.valueOf(df.format(bestQ1R))) + " " +
430                 puntosPorComas(String.valueOf(df.format(bestQ2R))) + "\t"
431                 );
432         printVector(rightShoulderQuantifier(numCols, bestQ1R, bestQ2R));
433 
434         System.out.print("Brute-force\tLinear\t");
435         System.out.print(
436                 puntosPorComas(String.valueOf(df.format(bestErrorL))) + "\t" +
437                 puntosPorComas(String.valueOf(df.format(bestQ1L))) + " " +
438                 puntosPorComas(String.valueOf(df.format(bestQ2L))) + "\t"
439                 );
440         printVector(linearQuantifier(numCols, bestQ1L, bestQ2L));
441 
442         System.out.print("Brute-force\tPower\t");
443         System.out.print(
444                 puntosPorComas(String.valueOf(df.format(bestErrorP))) + "\t" +
445                 puntosPorComas(String.valueOf(df.format(bestQ))) + "\t"
446             );
447         printVector(powerQuantifier(numCols, bestQ));
448 
449     }
450 
451 
452     // FORMA 2/3
453     public void owaMontecarlo (int numRepeticiones)
454     {
455         Random r = new Random();
456         double bestErrorRS = 100;
457         double bestErrorL = 100;
458         double bestErrorP = 100;
459         double bestQ1R = 0;
460         double bestQ2R = 0;
461         double bestQ1L = 0;
462         double bestQ2L = 0;
463         double bestQ = 0;
464         int numCols = filass[0].size();
465 
466         // 1. Cálculo de los errores
467 
468         for (int k=0; k < numRepeticiones; k++)
469         {
470             double q1 = r.nextDouble();
471             double q2 = r.nextDouble();
472             double q = 20 * q1;
473 
474             Double[] linear = linearQuantifier(filass[0].size(), q1, q2);
475             Double[] power = powerQuantifier(filass[0].size(), q1);
476 
477             Double errorMapeRS = null;
478             Double errorMapeL = null;
479             Double errorMapeP = null;
480 
481             if (q1 <= q2)
482             {
483                 Double[] rightShoulder = rightShoulderQuantifier(filass[0].size(), q1, q2);
484 
485                 //double resultadoRS = calculoOwaRS(filass[0], q1, q2);
486                 double errorAcumuladoRS = 0.0;
487                 for (int i = 0; i < numFilas; i++)
488                 {
489                     double resultadoRS = calculoOwaRS(filass[i], rightShoulder, q1, q2);
490                     errorAcumuladoRS += Math.abs((resultadoRS - datosParaComparar.get(i)) / resultadoRS);
491                 }
```

```
492                 errorMapeRS = (errorAcumuladoRS / numFilas) * 100;
493                 if (errorMapeRS < bestErrorRS)
494                 {
495                     bestErrorRS = errorMapeRS;
496                     bestQ1R = q1;
497                     bestQ2R = q2;
498                 }
499             }
500             else
501             {
502                 Double[] rightShoulder = rightShoulderQuantifier(filass[0].size(), q2, q1);
503 
504                 //double resultadoRS = calculoOwaRS(filass[0], q2, q1);
505                 double errorAcumuladoRS = 0.0;
506                 for (int i = 0; i < numFilas; i++)
507                 {
508                     double resultadoRS = calculoOwaRS(filass[i], rightShoulder, q2, q1);
509                     errorAcumuladoRS += Math.abs((resultadoRS - datosParaComparar.get(i)) / resultadoRS);
510                 }
511                 errorMapeRS = (errorAcumuladoRS / numFilas) * 100;
512                 if (errorMapeRS < bestErrorRS)
513                 {
514                     bestErrorRS = errorMapeRS;
515                     bestQ1R = q2;
516                     bestQ2R = q1;
517                 }
518             }
519 
520             if (q1 == 0)
521                 q2 = 0;
522             if (q1 == 1)
523                 q2 = 1;
524 
525             //double resultadoL = calculoOwaLinear(filass[0], q1, q2);
526             double errorAcumuladoL = 0.0;
527             for (int i = 0; i < numFilas; i++)
528             {
529                 double resultadoL = calculoOwaLinear(filass[i], linear, q1, q2);
530                 errorAcumuladoL += Math.abs((resultadoL - datosParaComparar.get(i)) / resultadoL);
531             }
532              errorMapeL = (errorAcumuladoL / numFilas) * 100;
533              if (errorMapeL < bestErrorL)
534              {
535                  bestErrorL = errorMapeL;
536                  bestQ1L = q1;
537                  bestQ2L = q2;
538              }
539 
540              //double resultadoP = calculoOwaPower(filass[0], q);
541              double errorAcumuladoP = 0.0;
542              for (int i = 0; i < numFilas; i++)
543              {
544                  double resultadoP = calculoOwaPower(filass[i], power, q);
545                  errorAcumuladoP += Math.abs((resultadoP - datosParaComparar.get(i)) / resultadoP);
546              }
547              errorMapeP = (errorAcumuladoP / numFilas) * 100;
548              if (errorMapeP < bestErrorP)
549              {
550                  bestErrorP = errorMapeP;
551                  bestQ = q;
552              }
553         }
```

```
555         // 2. Escritura de los resultados
556 
557         System.out.println();
558 //      System.out.println("***************\nOWA Brute Force\n");
559         System.out.print("Monte Carlo\tRight-shoulder\t");
560         System.out.print(
561                 puntosPorComas(String.valueOf(df.format(bestErrorRS))) + "\t" +
562                 puntosPorComas(String.valueOf(df.format(bestQ1R))) + " " +
563                 puntosPorComas(String.valueOf(df.format(bestQ2R))) + "\t"
564                 );
565         printVector(rightShoulderQuantifier(numCols, bestQ1R, bestQ2R));
566 
567         System.out.print("Monte Carlo\tLinear\t");
568         System.out.print(
569                 puntosPorComas(String.valueOf(df.format(bestErrorL))) + "\t" +
570                 puntosPorComas(String.valueOf(df.format(bestQ1L))) + " " +
571                 puntosPorComas(String.valueOf(df.format(bestQ2L))) + "\t"
572                 );
573         printVector(linearQuantifier(numCols, bestQ1L, bestQ2L));
574 
575         System.out.print("Monte Carlo\tPower\t");
576         System.out.print(
577                 puntosPorComas(String.valueOf(df.format(bestErrorP))) + "\t" +
578                 puntosPorComas(String.valueOf(df.format(bestQ))) + "\t"
579             );
580         printVector(powerQuantifier(numCols, bestQ));
581     }
582 
```

```java
    // FORMA 3/3
    public void MediaPonderada (int numRepeticiones)
    {
        // 1. Cálculo de los errores

        Random r = new Random();
        double bestError = 100;
        int numCols = filass[0].size();
        Double[] bestW = new Double[numCols];
        Double[] v = new Double[numCols];

        for (int k=0; k < numRepeticiones; k++)
        {
            for(int i = 0; i < numCols; i++)
                v[i] = r.nextDouble();

            //double res = weightedSum(v, filass[0]);
            double errorAcumulado = 0.0;
            for (int i = 0; i < numFilas; i++)
            {
                double res = weightedSum(v, filass[i]);
                errorAcumulado += Math.abs((res - datosParaComparar.get(i)) / res);
            }
            double errorMape = (errorAcumulado / numFilas) * 100;
            if (errorMape < bestError)
            {
                bestError = errorMape;
                for(int i = 0; i < numCols; i++)
                    bestW[i] = v[i];
            }
        }

        // 2. Escritura de los resultados

        System.out.println();
        System.out.println("***************\nMedia Ponderada\n");
        System.out.println(puntosPorComas("MAPE:\t" + String.valueOf(df.format(bestError))));
        printVector(bestW);

    }

    private static ArrayList<Double> lecturaDatosExcel (XSSFWorkbook workbook, FileInputStream entrada, int columnaSeleccionada)
    {
        ArrayList<Double> listaDevolver = new ArrayList<Double>();
        XSSFSheet sheet = workbook.getSheetAt(0);
        Iterator<Row> rowIterator = sheet.rowIterator();
        rowIterator.next();
        while (rowIterator.hasNext())
        {
            Row row = rowIterator.next();
            Iterator<Cell> cellIterator = row.cellIterator();

            while (cellIterator.hasNext())
            {
                Cell cell = cellIterator.next();
                if (cell.getColumnIndex() == columnaSeleccionada) { // Cambiar según el excel, pedir por teclado?

                    if(cell.getCellTypeEnum() == CellType.NUMERIC) {
                        double valor = cell.getNumericCellValue();
                        //System.out.println(cell.getRowIndex() + " " + valor );
                        listaDevolver.add(valor);
                    }
                }
            }
        }

        return listaDevolver;
    }


    //-----------------------------------------------------------------------------------

    public static String puntosPorComas (String s)
    {
        return s.replace(".",",");
    }

    //-----------------------------------------------------------------------------------
```

```java
    public static void main (String[] args)
    {

        Scanner scanner = new Scanner(System.in);

        System.out.println ("Introduzca sobre qué dataset quiere trabajar:");
        System.out.println ("   * Introduzca '1' si quiere trabajar con datos de temperaturas europeas del 2018");
        System.out.println ("   * Introduzca '2' si quiere trabajar con datos de temperaturas españolas entre 2014-2018");
        System.out.println ("   * Introduzca '3' si quiere trabajar con datos de turismo de Londres entre 2002-2019");
        int opcion = scanner.nextInt();

        int numeroEntradas=0, numeroMuestras=0, columnaInicial=0;
        File f = null;

        if (opcion == 1) {
            //f = new File ("1- TemperaturasPorMedias.xlsx");
            f = new File ("C:/Users/alvcr/Desktop/TFG/1- TemperaturasPorMedias.xlsx");
            //f = new File ("C:/Users/alvcr/Desktop/TFG/PruebaGeneralizando.xlsx");
            numeroEntradas = 5; //columnas
            numeroMuestras = 12; //filas
            columnaInicial = 3;
            //cambiar numeroEntradas=6 y columnaInicial=2 para contar Buenos Aires
        }
        else if (opcion == 2) {
            //f = new File ("2- Temperaturas2014-2018.xlsx");
            f = new File ("C:/Users/alvcr/Desktop/TFG/2- Temperaturas2014-2018.xlsx");
            numeroEntradas = 6; //columnas
            numeroMuestras = 60; //filas
            columnaInicial = 3;
        }
        else if (opcion == 3) {
            //f = new File ("3- TurismoLondres.xlsx");
            f = new File ("C:/Users/alvcr/Desktop/TFG/3- TurismoLondres (Total Visits).xlsx");
            //f = new File ("C:/Users/alvcr/Desktop/TFG/3- TurismoLondres (Total Nights).xlsx");
            //f = new File ("C:/Users/alvcr/Desktop/TFG/3- TurismoLondres (Total Spend).xlsx");
            numeroEntradas = 2; //columnas
            numeroMuestras = 72; //filas
            columnaInicial = 12;
        }


        else System.out.println ("Opción no válida");

         if (f.exists())
         {
             TRABAJO_FINAL tfg = new TRABAJO_FINAL (f, numeroEntradas, numeroMuestras, columnaInicial);

             long tiempoInicio = System.currentTimeMillis();
             tfg.owaBruteForceTabla(0.001);
             long tiempoTotal = System.currentTimeMillis() - tiempoInicio;
             System.out.println("Tiempo de proceso: " + tiempoTotal / 1000. + " segundos.");

             tiempoInicio = System.currentTimeMillis();
             tfg.owaBruteForce(0.0001);
             tiempoTotal = System.currentTimeMillis() - tiempoInicio;
             System.out.println("\nTiempo de proceso BruteForce (0.0001): " + tiempoTotal / 1000. + " segundos.");

             tiempoInicio = System.currentTimeMillis();
             tfg.owaBruteForce(0.001);
             tiempoTotal = System.currentTimeMillis() - tiempoInicio;
             System.out.println("\nTiempo de proceso BruteForce (0.001): " + tiempoTotal / 1000. + " segundos.");

             tiempoInicio = System.currentTimeMillis();
             tfg.owaMontecarlo(500000);
             tiempoTotal = System.currentTimeMillis() - tiempoInicio;
             System.out.println("\nTiempo de proceso Montecarlo (500000): " + tiempoTotal / 1000. + " segundos.");
/*
             tiempoInicio = System.currentTimeMillis();
             tfg.MediaPonderada(1000000);
             tiempoTotal = System.currentTimeMillis() - tiempoInicio;
             System.out.println("\nTiempo de proceso: " + tiempoTotal / 1000. + " segundos.");
*/
         }
         else
             System.out.println ("El archivo indicado no existe");

         scanner.close();
    }

}

```