Proyecto de Minería de Datos:
Borrador de memoria completa

Markus Fischer • Guzmán López • David Pérez • Ander Raso

Índice

1. Introducción

1.1. Objetivo de la tarea

En este proyecto se nos encarga la tarea de trabajar en el campo del *Text Mining* y del *clustering* de documentos. Nuestro objetivo consiste en realizar una clasificación no supervisada de una gran cantidad de textos. La clasificación no supervisada consiste en agrupar las instancias en distintos grupos, o clusters, según su similaridad. Se intenta conseguir que las instancias de un cluster determinado sean similares entre sí a la vez que distintas de las instancias que no pertenecen al cluster.

1.2. Propuesta de trabajo del grupo

Como grupo, hemos decidido realizar la tarea sobre la colección de autopsias verbales que se propuso como opción. Esto se debe a que, ademas de considerar el tema muy interesante, creemos que es una propuesta que se encuentra muy próxima al uso que se da al *Text Mining* en el ámbito científico.

Sin embargo, de esta decisión también surgen ciertos retos a los que debemos poner solución:

- Muchos de los reportes de autopsia están en un lenguaje poco preciso y muchas veces ininteligible, probablemente causado tanto por el desconocimiento de los que dieron el reporte, como por las traducciones que se han hecho a estos.
- En muchas ocasiones no hay reporte verbal o este es irrelevante, por lo que la única información útil de que se dispone es de los datos del difunto, tales como país, edad, sexo, etc.

2. Descripción y análisis de datos

En este proyecto vamos a trabajar con autopsias verbales. Estas son reportes que dan familiares o personas cercanas a un fallecido en lugares donde, por norma general, no se realiza una autopsia post-mortem a menos que sea estrictamente necesario. Estas autopsias se componen de información básica del paciente (edad, sexo, etc.) así como del reporte oral que ha dado el relativo al que se ha entrevistado.

La base de datos de las autopsias verbales se compone de casi 12000 instancias. En cada instancia disponemos de una serie de atributos:

- **newid:** Identificador numérico de la instancia.
- **module:** Grupo de edad del fallecido. Los valores posibles son “neonate”, “child” y “adult” (Fig: 1).
- **site:** Lugar del que proviene la autopsia: los valores posibles para este campo son (Fig: 2):
  - AP=Andhra Pradesh, India

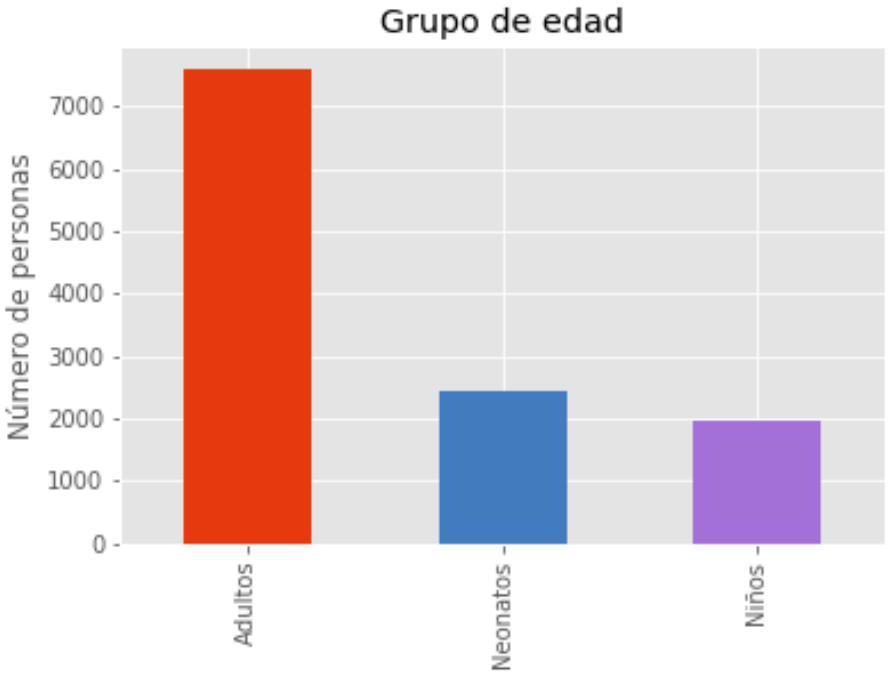

Figura 1: Número de instancias según la edad del fallecido

  - Dar=Dar es Salaam, Tanzania
  - UP=Uttar Pradesh, India
  - Pemba=Pemba, Tanzania
  - Bohol=Bohol, Philippines
  - Mexico=Distrito Federal, Mexico

- **gs_text34:** Causa de la muerte del paciente.
- **sex:** Sexo del difunto. Los valores posibles son 1, 2, 3 y 4. Cada uno de estos valores hace referencia a "hombre", "mujer", "se niega a responder" y "desconocido" respectivamente (Fig: 3).
- **age_years:** Edad del fallecido en años. En caso de ser igual a 999 significa que es desconocida.
- **age_months:** Edad del fallecido en meses. En caso de ser igual a 99 significa que es desconocida.
- **age_days:** Edad del fallecido en días. En caso de ser igual a 99 significa que es desconocida.
- **open_response:** Declaración de la persona cercana al difunto en caso de que hubiese una. En este texto se han eliminado todas las referencias que se diesen en la declaración que pudiesen relacionarse con el difunto para asegurar su privacidad, tales como menciones a los lugares y hospitales

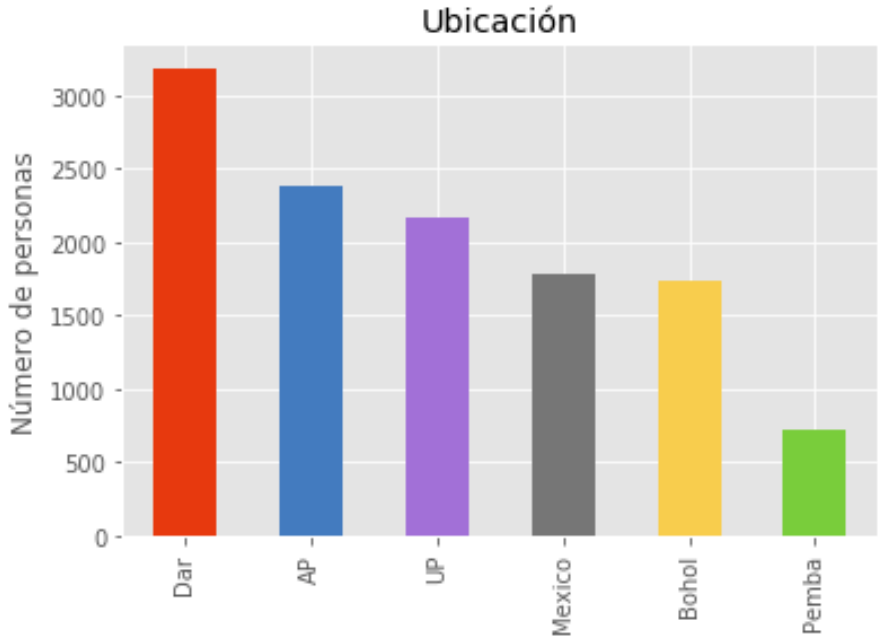

Figura 2: Número de instancias según la ubicación del fallecido

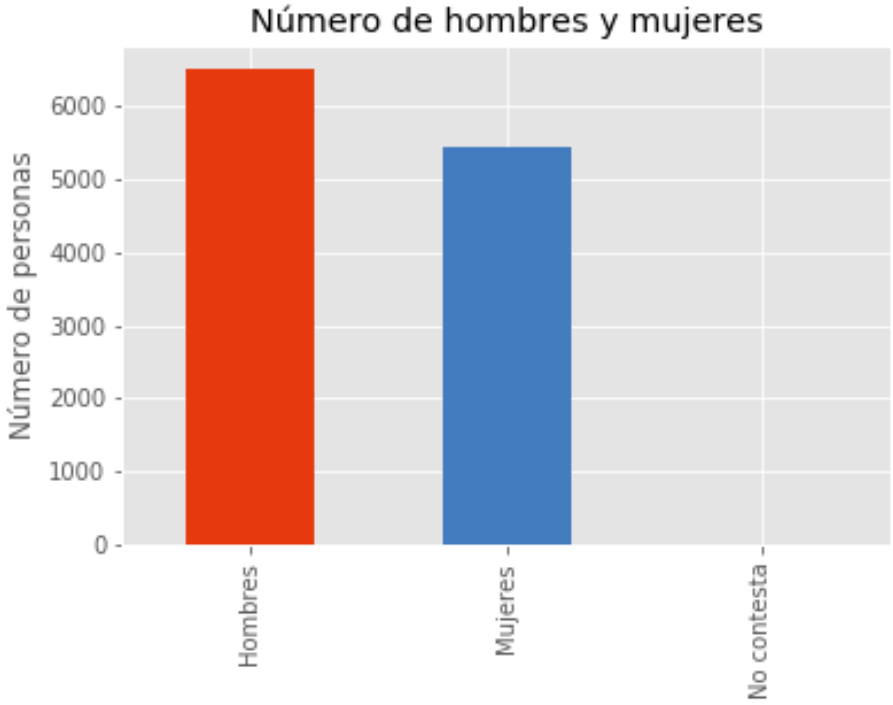

Figura 3: Número de instancias según el sexo del fallecido

en los que se ha encontrado, el nombre de personas o fechas concretas en relación al fallecido, etc. En su lugar se han sustituido por palabras neutrales entre corchetes, tales como "[PERSON]" o "[HOSPITAL]".

3. Preproceso de datos

El formato original del archivo de autopsias es un xlsx, es decir, una hoja de cálculo de Microsoft Office. Esto nos facilita mucho la tarea de procesar los textos, ya que es muy fácil convertir de xlsx a formatos de texto plano. En nuestro caso elegimos csv, ya que es un formato muy limpio, fácil de entender y compatible con gran parte de las librerías que vamos a utilizar.

3.1. Limpeza de textos

Una vez tenemos nuestro documento como csv, procedemos a preprocesarlo. Ya que los valores de identificador, grupo de edad, lugar, diagnóstico, sexo y edad están codificados según unos criterios que se mantienen a lo largo de todas las instancias, sabemos que no debemos preprocesarlos. Sin embargo, en el campo de la respuesta verbal sí que tenemos que solucionar algunas anomalías:

- **Mayúsculas y minúsculas:** Para evitar diferencias entre palabras escritas completamente en minúscula, completamente en mayúscula o con la primera letra mayúscula, debemos transformar todas ellas a un mismo formato. En nuestro caso el formato escogido son las minúsculas.
- **Saltos de línea:** En algunos casos nos encontramos con saltos de línea internos en el texto. Ya que csv separa sus diferentes atributos por comas y las instancias por líneas, los saltos de línea internos en el texto crean inconsistencias y errores al procesar el csv. El método más fácil para deshacerse de ellos pero mantener el texto en el mismo formato es sustituirlos por espacios.
- **Números:** Los números, pese a ser uno de los elementos más comunes en los textos, han sido uno de los más difíciles de preprocesar, no por su dificultad en cuanto a programar, sino a la decisión que tomar en cuanto a ellos. Esto se debe a que se toman como una palabra por si mismos y pese a tener un peso informativo alto, sólo es así cuando va acompañado de otra palabra a la que calificar. Ya que los programas de *Text Mining* los van a considerar independientes del resto de palabras, hemos considerado que la información que van a aportar es suficientemente baja como para simplemente eliminarlos del texto.
- **Tokens:** Se consideran tokens a todas las palabras recurrentes que no tienen ni significado ni peso informativo, como por ejemplo "the" o "a". Mediante un diccionario de tokens en la lengua inglesa hemos podido detectar todos ellos y eliminarlos del texto.
- **Símbolos:** Aquí tenemos varios problemas que debemos arreglar de diferentes formas.

- **Barras "/":** En ocasiones nos encontramos con dos palabras escritas de modo "palabra1/palabra2". Para evitar que se procesen ambas como una misma palabra, realizamos el mismo procedimiento que utilizamos con los saltos de línea: cambiamos las barras por espacios.
- **Corchetes "[ ]":** En gran parte de las instancias que disponen de autopsia verbal nos encontramos con referencias a nombres de personas, hospitales, años concretos, etc. Para preservar la privacidad de los fallecidos y sus relativos estos han sido sustituidos por sus correspondientes palabras clave entre corchetes, como por ejemplo "[PATIENT]" o "[HOSPITAL]". Ya que estos datos son en la inmensa mayoría de casos de poca utilidad, hemos decidido que estás palabras serán completamente eliminadas de los textos.

3.2. *Word vector*

Después de limpiar los textos de las instancias necesitamos convertir ese texto a un formato que podamos utilizar. El formato escogido es el TF-IDF, el cual consiste en una representación vectorial del conjunto de palabras presentes en todos los documentos (instancias). Cada una de las palabras se evalúa individualmente y se le asigna un valor numérico en función al número de apariciones que tiene en cada documento, pero teniendo en cuenta también en cuántos de los documentos aparece para evitar darle excesiva importancia a palabras que simplemente se repiten por ser comunes.

Escogimos la representación TF-IDF en favor de *Bag of Words* por el hecho de que aporta más información. *Bag of Words* se limita a indicar la presencia o ausencia de palabras en los documentos mientras que TF-IDF le asigna un valor en función de su aparente importancia teniendo en cuenta todos los documentos.

4. Clustering

Para el *clustering* hemos decidido utilizar el algoritmo *k-means clustering*, el cual implementaremos en Python. Algunas de las opciones para el algoritmo a tener en cuenta son:

- **Inicialización de los centroides:** Hay varias formas de elegir los valores iniciales de los centroides, y su inicialización podría tener repercusíon en los resultados.
  - **Inicialización a instancias aleatorias:** cada centroide se toma el valor de una instancia elegia aleatoriamente. En nuestro caso los inicializamos de esta manera.
  - **Inicialización a partir de un *clustering* previo:** Se realiza una agrupación preeliminar a $2k$ clusters. De esos clusters se eligen los k que más separados estén para la inicialización.
- **Cálculo de las distancias:** La fórmula utilizada para calcular la distancia será la distancia Minkowski, con $m \in \Re$. $m = 1$ es la distancia

Manhattan y $m = 2$ es la distancia Euclidea. Cálculo de la distancia con Minkowski: [2]

$$D(X, Y) = (\sum_{i=1}^{n} |x_i - y_i|^m)^{\frac{1}{m}} \quad (1)$$

D: distancia
x, y: vectores
m: orden (normalmente 1 o 2 para las distancias Manhattan o Euclídea respectivamente)
En el caso concreto del cálculo de la distancia segun Manhattan ($m = 1$)[2]:

$$D(X, Y) = \sum_{i=1}^{n} |x_i - y_i| \quad (2)$$

- **Definición de distancia *inter-cluster*:** También existen varias formas para buscar la distancia entre dos clusters distintos. En este caso, vamos a trabajar con dos:

  - *Complete-link*. La distancia entre dos clusters es igual a la distancia entre las dos instancias, una de cada cluster, más lejanas entre sí.

$$d(C_i, C_j) = \max_{\forall x^{(r)} \in C_i; \forall x^{(s)} \in C_j} d(x^{(r)}, x^{(s)}) \quad (3)$$

  - *Single-link*. La distancia d entre dos clusters (C_i, C_j) es igual a la distancia entre las dos instancias ($x^{(r)}$,$x^{(s)}$), una de cada cluster, más cercanas entre sí.

$$d(C_i, C_j) = \min_{\forall x^{(r)} \in C_i; \forall x^{(s)} \in C_j} d(x^{(r)}, x^{(s)}) \quad (4)$$

  - *Average-link*. La distancia entre dos clusters (C_i, C_j) es igual a la distancia entre los centroides (m_i,m_j) de cada cluster.

$$d(C_i, C_j) = d(m_i, mj) \text{ para } m_i = \frac{1}{|C_i|} \sum_{\forall x^{(r)} \in C_i} x^{(r)} \quad (5)$$

  [3]

- **Criterio de convergencia:** Parámetro para decidir cuándo dar por finalizado el algoritmo de *clustering*. Terminamos éste cuando la variación de los centroides sea más pequeña que nuestro umbral de una iteración a la siguiente. También tenemos un número máximo de iteraciones tras las cuales el algortimo termina forzosamente, para evitar posibles ciclos.

4.1. Representación gráfica

Una vez que tenemos las instancias separadas en clusters, sería interesante representarlas gráficamente para poder ver como se relacionan los clusters, calculados a partir del texto describiendo la enfermedad, con la causa de la muerte anotada en las autopsias. Para ello, tenemos que reducir el enorme espacio de

input: Conjunto de instancias X

```
// inicializar los k centroides
```
inicializar $m_i, i \in [1, k]$;
$terminado \leftarrow False$;
while *no terminado* **do**
    `// para cada instancia, activar el bit correspondiente al centroide más cercano`
    foreach $x^t \in$ X **do**
        if $\|x^t - m_i\| = min_j \|x^t - m_i\|$ **then**
            $b_i^t \leftarrow 1$;
        else
            $b_i^t \leftarrow 0$;
        end
    end
    `// guardar los centroides anteriores para comparación`
    $n \leftarrow m$;
    `// actualizar los nuevos centroides como la media de todas sus instancias`
    foreach $m_i, i \in [1, k]$ **do**
        $m_i \leftarrow \sum_t b_i^t x^t / \sum_t b_i^t$;
    end
    `// comprobar convergencia de los centroides`
    if $\|n - m\| < threshold$ **then**
        $terminado \leftarrow True$;
    end
end

Algorithm 1: Algoritmo *k-means clustering* (pseudocódigo)

atributos de las instancias a algo que podamos representar en un espacio de no más de tres dimensiones. Para ello pensamos en utilizar el método PCA (*Principal Components*) el cual reduce un gran número de variables a una serie de valores representativos, ortogonales entre sí para representarlos en espacios de pocas dimensiones. Con este método podemos convertir nuestras instancias con cerca de 8000 atributos a dos atributos cada una para poder representarlas en un plano cartesiano.

5. Evaluación

Para evaluar el modelo obtenido vamos a utilizar la métrica de SSE (*Sum of Square Errors*). Para cada uno de los clusters calculamos el error (distancia) cuadrático de cada instancia con respecto al centroide, y sumando todos estas distancias obtenemos el SSE total del cluster; su cohesión.

$$SSE(C_i) = \sum_{x \in C_i} d^2(x, c_i) \tag{6}$$

De la misma forma, sumando el SSE de cada cluster podemos obtener la cohesión total de la partición.

$$C = \{C_1, C_2, C_3, ..., C_k\}$$

$$SSE(C) = \sum_{i=1}^{k} SSE(C_i) = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i) \tag{7}$$

Para comparar cohesión entre clusters, calcularemos primero el SSE medio de los clusters, dividiendo el SSE de cada uno entre el número de instancias que contiene, ya que los clusters más poblados tendrán, por lo general, un SSE mayor solamente por su alto número de instancias, independientemente de su cohesión.

6. Experimentos

Como experimentos vamos a realizar el clustering con distintos valores en los principales parámetros: k (número de clusters a crear), m (parámetro para la distancia Minkowski) y la estrategia de inicialización de centroides.

6.1. Resultados

6.1.1. Parámetro k

Hemos probado a hacer el clustering con dos valores distintos de k: 48 y 96. En los datos hay un total de 46 causas de muerte registradas por lo que decidimos crear, por un lado, tantos clusters como clases, y, por otro, el doble, para ver si había alguna diferencia notable.

No resultó haber gran diferencia en los SSE medios de los clusters, pero mirando a los heatmaps de las particiones (Figs 4 y 5) se puede notar como la relación entre cluster y causa de muerte está bastante más dispersa cuando tenemos solo 48 clusters que cuando tenemos 96.

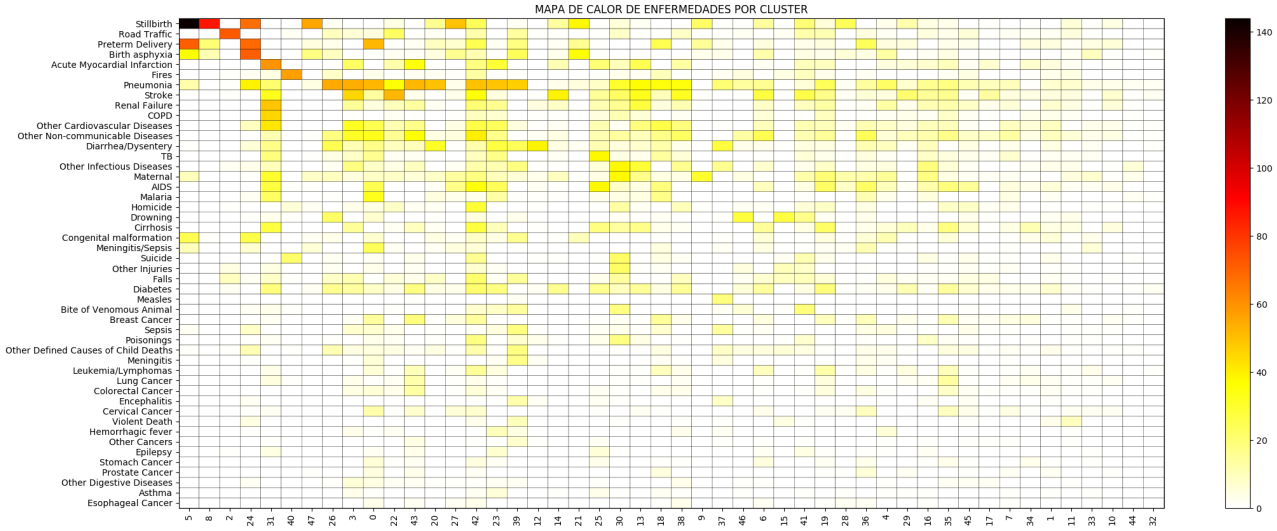

Figura 4: Heatmap (k=48, m=2, init=random)

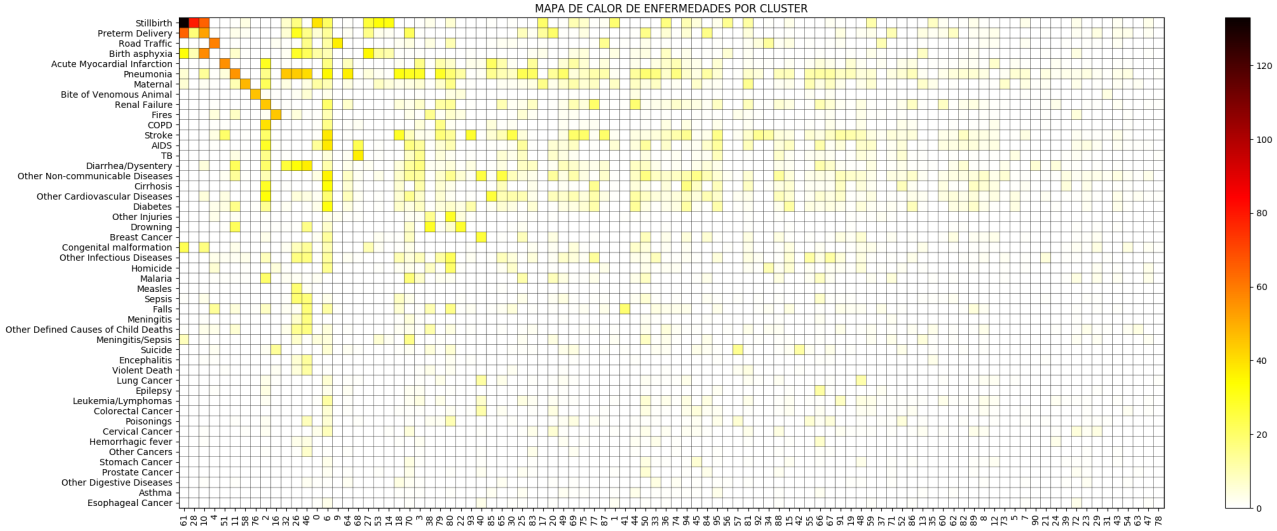

Figura 5: Heatmap (k=96, m=2, init=random)

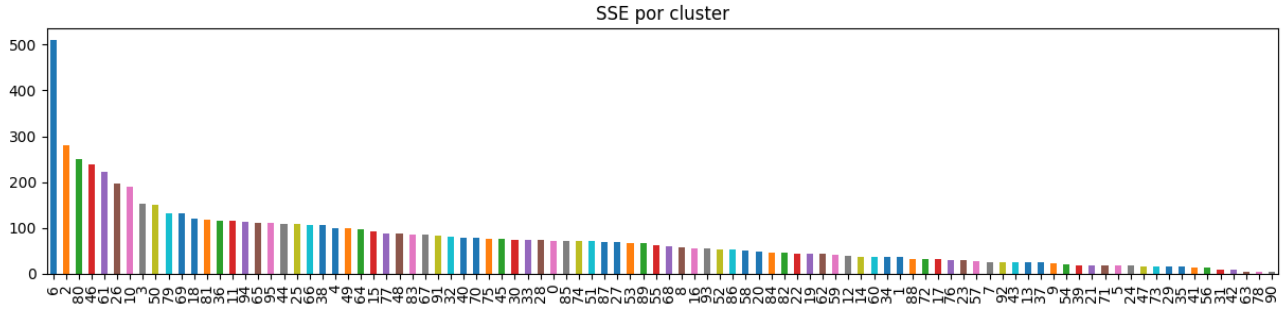

Figura 6: SSE por cluster (k=96, m=2, init=random)

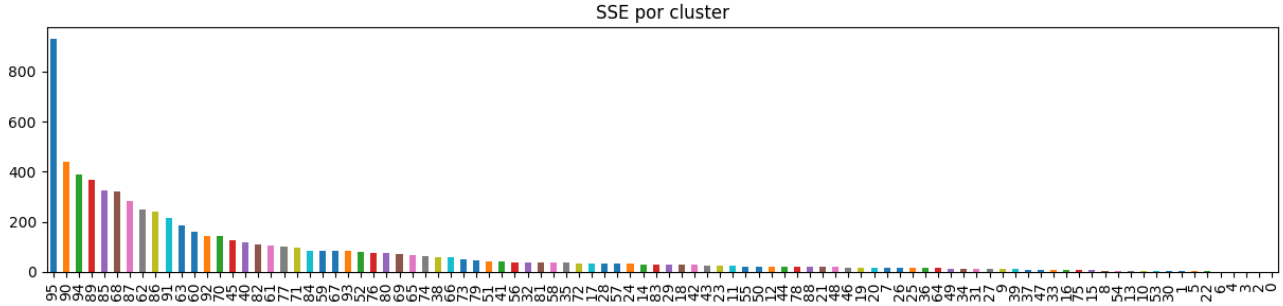

Figura 7: SSE por cluster (k=96, m=2, init=2k)

6.1.2. Inicialización de centroides

Los dos métodos de inicialización que hemos probado son, inicialización aleatoria e inicialización a partir de una partición anterior de 2k clusters, de los cuales se eligen los k clusters más separados entre sí.

Comparando los resultados, para los dos casos el SSE medio por cluster es similar, sin embargo el SSE total por cluster es considerablemente distinto. En la inicialización aleatoria los clusters con mayor SSE tiene menos error que sus contrarios, mientras que en el otro extremo, los clusters de inicialización aleatoria con menor SSE tienen más error que los de inicialización a partir de partición previa. Esto se debe a que la densidad de instancias está más centralizada en la inicialización 2k que en la aleatoria; hay más clusters con muchas instancias y más clusters con pocas.

Parece que al iniciar el clustering con centroides más separados entre sí, las distribución de instancias ha sido menos uniforme, haciendo que algunos clusers sean más densos y que otros lo sean menos en comparación a la inicialización aleatoria. Por esta razón, la relación entre los clusters y las clases está menos definida. Esto también se ve comparando el heatmap de la inicialización 2k (Fig 8) con el de la inicialiazción aleatoria (Fig 5).

Figura 8: Heatmap (k=96, m=2, init=2k)

6.1.3. Distancia Minkowski

Por último, probamos a variar el parámetro m de la distancia Minkowski y elegimos los valores 2 y 4. El resultado de utilizar $m = 4$ es un el error total de los clusters (Fig 9) lígeramente más alto en comparación a la misma operación realizada con $m = 2$ (Fig 6). Sin embargo, comparando los mapas de calor (Figs 5 y 10), parece que $m = 4$ hace un trabajo ligeramente mejor de segregar las instancias según su clase.

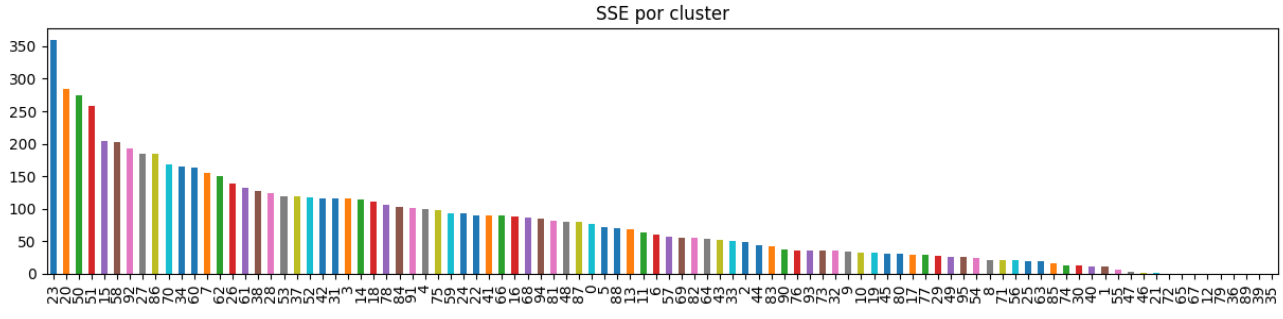

Figura 9: SSE por cluster(k=96, m=4, init=random)

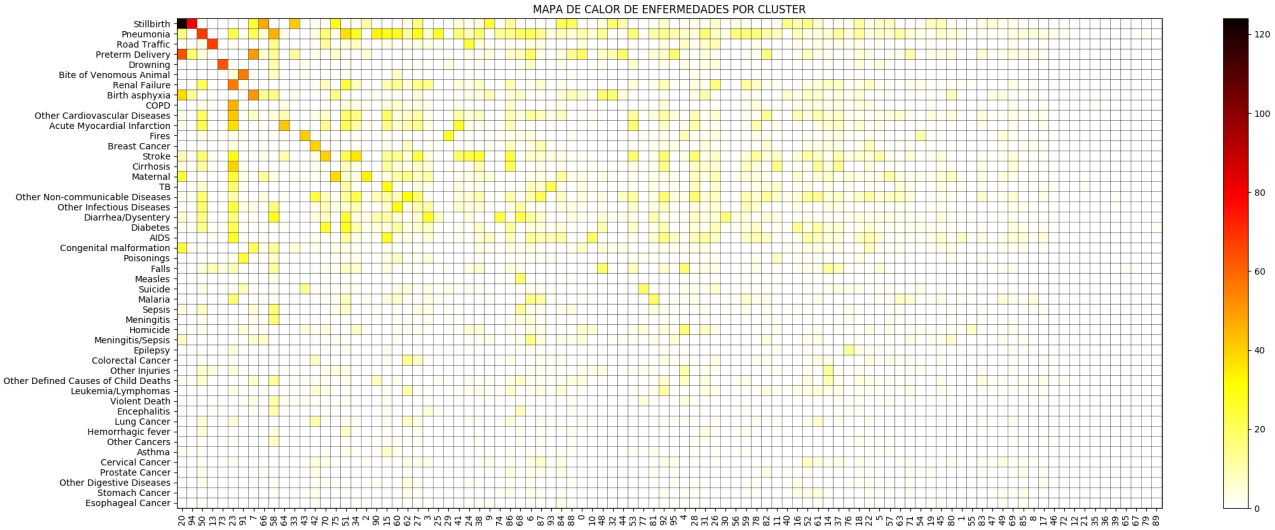

Figura 10: Heatmap (k=96, m=4, init=random)

7. Conclusiones

Gracias a este proyecto hemos podido ver de primera mano una de las posibles aplicaciones de las técnicas de clustering. Hemos de tener en cuenta que es una técnica ampliamente utilizada en el mundo empresarial y laboral a unas escalas que no son comparables a nuestro pequeño proyecto, pero éste sirve como una buena aproximación a un mundo que todavía está en auge y en el que aún no hay muchas normas impuestas.

En cuanto a los resultados del proyecto, ...

En general nuestro programa cumple satisfactoriamente con los objetivos propuestos. Hemos sido capaces de desarrollarlo de forma que sea lo más potente, versátil y eficiente posible. Han sido estos dos últimos de especial importancia para nuestro grupo, puesto que por un lado la versatilidad nos ha permitido realizar pruebas con diferentes escenarios y parámetros para poder buscar fácilmente la mejor opción de clustering, y por otro lado hemos tenido que hacer nuestro código lo más eficiente posible ya que ha habido que realizar una gran cantidad de operaciones sobre un conjunto de datos de por si bastantegrande, y en las primeras iteraciones del programa los ordenadores menos potentes del grupo no eran capaces de realizar estas tareas correctamente. Esto fue corregido posteriormente, consiguiendo que con el código final se pueda realizar la tarea en tiempos bastante bajos para la magnitud de ésta.

Sin embargo, si tuviésemos que mejorar nuestro software una de las cosas en las que podríamos centrarnos es en dividir las instancias de forma previa

al clustering en función de ciertos parámetros que hacen que estas instancias no sean completamente homogeneas. Un claro ejemplo de esto son los rangos de edad con los que están etiquetadas las instancias, en los que podemos ver que para aquellas instancias catalogadas como neonatos el abanico de causas de muerte es muy reducido. Otro posible ejemplo es el sexo de cada una de las personas, ya que es posible que no se pueda padecer alguna enfermedad según éste (un claro caso sería un cancer testicular), o simplemente influir mucho en el conjunto de enfermedades que le pueden afectar. En ambos casos podría ser muy interesante disponer de la funcionalidad de realizar un clustering por separado.

Otro de las posibles áreas de mejora tiene que ver con los parámetros que nuestro algoritmo utiliza, tales como el número de clusters, el orden de la distancia de Minkowski, o la inicialización de los clusters. En este momento todos estos valores son introducidos por el usuario y, si bien eso añade versatilidad a nuestro programa, podría ser el propio programa el que ofreciese la posibilidad de probar con diferentes valores y ofrecer al usuario la mejor solución que pueda encontrar.

8. Bibliografía

[1] N.N., Wikipedia, 30 Octubre 2018, https://en.wikipedia.org/wiki/Tf-idf

[2] Binbin Lu et Al., The Minkowski approach for choosing the distance metric in geographically weighted regression, International Journal of Geographical Information Science, 2015

[3] Alicia Pérez, Tema 2: CLUSTERING, UPV/EHU Bilbao, 10 Septiembre 2018