UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Machine Learning: Examen 1. Métodos de clustering y análisis de componentes principales.

Alumno:
Ing. Enrique Mena Camilo

Profesor:
Dr. Marco Antonio Aceves Fernández

Mayo 2023

Índice

1. Objetivos

Implementar y comparar dos algoritmos de agrupamiento diferentes, K-means y Affinity, utilizando como datos a clasificar la base de datos de desempeño de deportistas.

2. Introducción

Los problemas de agrupamiento probablemente son uno de los temas más estudiados dentro del aprendizaje máquina y la minería de datos. El agrupamiento de datos tiene múltiples dominios de aplicación como en datos multimedia, de texto, redes sociales, datos biológicos y más.

En este trabajo se aplica dicha técnica de agrupamiento a la base de datos bodyPerformance, la cual contiene información del desemepeño físico de atletas. El agrupamiento de la ya mencionada base de datos se realizará con dos algoritmos distintos, K-means y el algoritmo Affinity. En el presente trabajo se desarrollan las funciones, en lenguaje de Python, necesarias para el agrupamiento de cualquier base de datos, sin embargo, toma como referencia la base de datos utilizada en este trabajo.

3. Marco teórico

El agrupamiento o clustering (en inglés) es una herramienta, específica del aprendizaje no supervisado, encargada de agrupar un conjunto de objetos sin etiqueta en clases o clústeres que son altamente similares entre sí, es decir, comparten propiedades y/o características, mientras que son escasamente diferente al resto de objetos, dado que no tienen propiedades y/o características similares. Básicamente, se desconoce si los datos ocultan patrones, así que el algoritmo se encarga de encontrarlos si es que los hay.

El agrupamiento tiene una gran cantidad de aplicaciones en distintos ámbitos de la vida, desde segmentación de mercado hasta en imágenes médicas, así como en análisis de redes sociales, agrupación de resultados de búsqueda, segmentación de imágenes, motores de búsqueda y más.

3.1. K-Means

El algoritmo k-means es un método de clustering basado en particiones que divide un conjunto de datos en k clusters de manera iterativa. El objetivo del algoritmo es minimizar la suma de los cuadrados de las distancias entre cada punto de datos y el centroide de su cluster asignado.

El algoritmo se puede describir en los siguientes pasos:

1. Seleccionar k centroides iniciales aleatorios del conjunto de datos.

2. Asignar cada punto de datos al cluster cuyo centroide esté más cercano.

3. Recalcular el centroide de cada cluster como la media de todos los puntos de datos asignados a él.

4. Repetir los pasos 2 y 3 hasta que la asignación de los clusters ya no cambie o se alcance un número máximo de iteraciones.

La distancia euclidiana cuadrática se utiliza para medir la distancia entre cada punto de datos y los centroides. Esta distancia se define como $d(x_i, c_j)^2 = \sum_{l=1}^{m}(x_{i,l} - c_{j,l})^2$, donde m es la dimensión del espacio de características.

Las ventajas del algoritmo k-means son que es fácil de entender e implementar, computacionalmente eficiente para conjuntos de datos grandes y adecuado para conjuntos de datos con una estructura clara de clusters.

Sin embargo, el algoritmo k-means también tiene algunas desventajas, como la elección inicial de los centroides que puede afectar significativamente los resultados, la necesidad de conocer el número de clusters de antemano y la falta de garantía de que el algoritmo encuentre la solución óptima debido a la aleatoriedad en la selección inicial de los centroides.

3.2. Affinity Propagation

El algoritmo Affinity Propagation es un método de clustering basado en la propagación de afinidad, que es una medida de la similitud entre dos puntos de datos. En lugar de agrupar los puntos de datos en clusters explícitos, el algoritmo encuentra un conjunto de puntos de datos llamados .^exemplars"que mejor representan el conjunto de datos.

El algoritmo se puede describir en los siguientes pasos:

1. Inicializar la matriz de similitud S entre los puntos de datos.
2. Inicializar la matriz de responsabilidad R entre los puntos de datos.
3. Inicializar la matriz de disponibilidad A entre los puntos de datos.
4. Actualizar la matriz de responsabilidad utilizando la siguiente ecuación: $R_{i,k} \leftarrow S_{i,k} - \max_{k' \neq k}(A_{i,k'} + S_{i,k'})$, donde i y k son índices de puntos de datos y $R_{i,k}$ es la responsabilidad del punto i con respecto al punto k.
5. Actualizar la matriz de disponibilidad utilizando la siguiente ecuación: $A_{i,k} \leftarrow \min(0, R_{k,k} + \sum_{i' \neq i, i' \neq k} \max(0, R_{i',k}))$, donde $A_{i,k}$ es la disponibilidad del punto i con respecto al punto k.
6. Actualizar los exemplars para cada punto de datos utilizando la siguiente ecuación: $s(i) \leftarrow \arg\max_k(R_{i,k} + A_{i,k})$, donde $s(i)$ es el índice del exemplar del punto i.
7. Repetir los pasos 4 a 6 hasta que la matriz de disponibilidad converja o se alcance un número máximo de iteraciones.

   El algoritmo utiliza una medida de similitud entre dos puntos de datos, que puede ser una medida de distancia o similitud como la distancia euclidiana, la similitud coseno o una medida de correlación.

   Una vez que el algoritmo ha convergido, cada punto de datos se asigna a su exemplar correspondiente, que representa el cluster al que pertenece el punto de datos.

   Las ventajas del algoritmo Affinity Propagation son que no requiere conocer el número de clusters de antemano, puede manejar conjuntos de datos de alta dimensionalidad y es robusto a los datos ruidosos y atípicos.

   Sin embargo, el algoritmo también tiene algunas desventajas, como su sensibilidad a la elección de los parámetros, su computacionalmente costosa complejidad de tiempo y la falta de garantía de que la solución encontrada es la óptima global.

4. Materiales y métodos

Para el desarrollo de este examen se utilizó el lenguaje de programación Python en su versión 3.10, con el que se diseñó un conjunto de scripts y una libreta de Jupyter para cumplir los objetivos de la práctica.

4.1. Conjunto de datos utilizado

El conjunto de datos utilizado para el desarrollo de este examen consta de información relacionada al desempeño físico de un conjunto de atletas. Dicho conjunto de datos se encuentra disponible para su acceso público mediante la plataforma Kaggle.

Dicho conjunto de datos consta de un total de 11 atributos y variable objetivo, así como un total de 13,393 instancias sin presencia de datos faltantes y en balanceadas en porciones del 25 %. Los tipos de datos disponibles en el conjunto de datos mencionado anteriormente constan de:

- Ordinales
  - age
- Continuos
  - height_cm
  - weight_kg
  - body fat_%
  - diastolic
  - systolic
  - gripForce
  - sit and bend forward_cm
  - situps counts
  - broad jump_cm
- Categóricos
  - gender
  - class (variable objetivo)

4.2. Pre-procesamiento de datos

4.2.1. Codificación de datos

Los datos disponibles en el atributo *gender* y en la variable objetivo se sometieron a un proceso de codificación, esto debido a que originalmente se encontraban en formato de texto y ya se había identificado que eran valores de tipo categórico.

4.2.2. Normalización de datos

Se realizó un proceso de obtención de la distribución de los datos que conforman el conjunto de datos, usando histogramas y diagramas de cajas. Se pudo observar que para la mayoría de los casos se contaba con una distribución del tipo uniforme, por lo que se usó el método *z-score* para normalizar los datos.

Para el caso específico del atributo *age*, se optó por utilizar el método de normaliación min-max, ya que no segía la distribución normal. Otro caso particular fue el observado en el atributo *gender*, el cual se encontraba con únicamente 2 observaciones, por lo cuál no fue necesario aplicar algún método de normalización. Por último, el caso más llamativo se presentó con el atributo *grip_force*, el cual mostraba una distribucion bi-modal; para este último caso se optó por seguir usando el método *z-score*, ya que será de utilidad tener a todos los atributos dentro del mismo rango para lograr un desempeño adecuado en el algoritmo de análisis de componente principales.

4.3. Análisis de componentes principales

Para la implementación del algoritmo de análisis de componente princiáles (PCA) se siguieron dos aproximaciones que serán mencionadas a continuación.

4.3.1. Extracción de atributos más relevantes

En esta aproximación se buscó obtener al grupo de atributos que, en conjunto, lograran maximizar la relevancia de los datos. Esta aproximación se realizó mediante la obtención de los valores y vectores propios del conjunto de datos, para posteriormente calcular su co-varianza y extraear a los de mayor relevancia.

4.3.2. Obtención de subespacio de componentes principales

El procedimento para la obtención de un subespacio de componentes es relativamente facil, se sigue un procedimiento similar al descrito en la sección anterior, teniendo el diferenciador de crear una matriz de proyección y con esta generar el nuevo sub-espacio de componentes principales.

4.4. Métodos de clustering

Para este trabajo se implementaron dos métodos de clustering, uno con la flexibilidad de poder seleccionar la cantidad de clusters que se desea estimar, mientras que el segundo realiza un procedimiento de selección completamente automático.

4.4.1. K-means

El método k-means se implementó siguiendo los pasos descritos en el marco teórico, tomando como apoyo a los métodos disponibles en la librería Numpy para agilizar las operaciones matriciales a desarrollar. Este algoritmo fue probado en dos epacios, el primero consitió de solamente generar los clusters y etiquetas usando todas las instancias de los componentes princpialoes, mientras que el segundo se utilizó para ser evaluado mediante validación cruzada.

4.4.2. Affinity propagation

El método affinity propagation fue implementado usando la librería scikit-learn, la cual cuenta con métodos que permiten agilizar el desarrollo de aplicaciones de clustering. Este método se implementó dentro de una función que permite realizar el entrenamiento con un único llamado, lo cuál a su vez favorece su desempeño.

4.5. Evaluación con validación cruzada

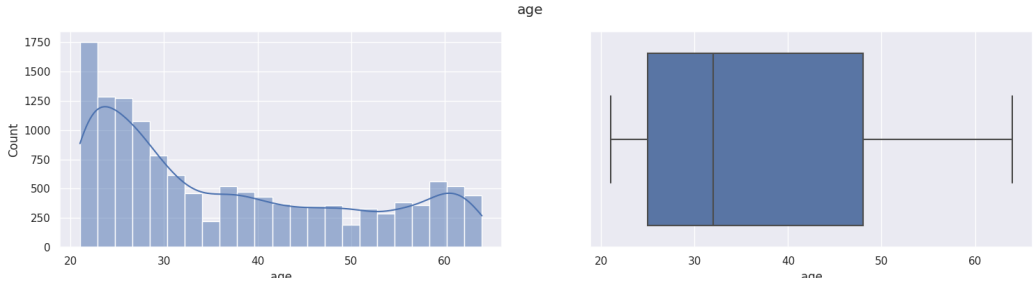
Para la evaluación mediante validación cruzada se optó por utilizar la métrica de exactitud, la cual proporciona un índice de la calidad de la clasificación del algoritmo. Para la obtención de esta métrica se tomó como referencia un método propuesto en [1] que muestra como lidiar con la aleatoriedad de los clusters.

Se implementó una validación cruzada de 10 segmentos, donde al finalizar el proceso de validación se obtuvo el valor de exactitud promedio y un diagrama de caja.

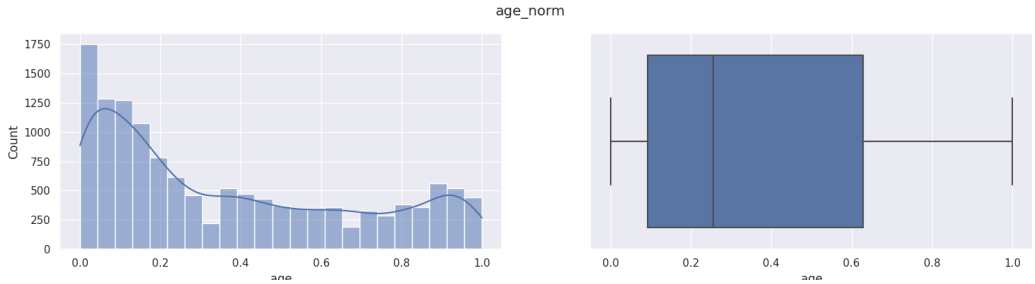
5. Resultados

5.1. Normalización de datos

Tomando como ejemplo a los casos especiales para el conjunto de datos utilizado, se puede observar en la Figura 2 que no existen cambio alguno en la distribución de los datos del atributo *age*, mientras que la Figura **??** muestra que la elección del método *z-score* para la normalización de datos continuos fue acertada.

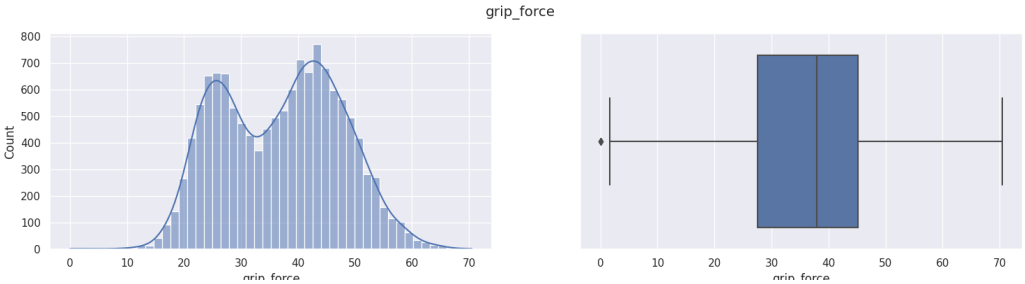


(a) Distribución previa a normalizar.

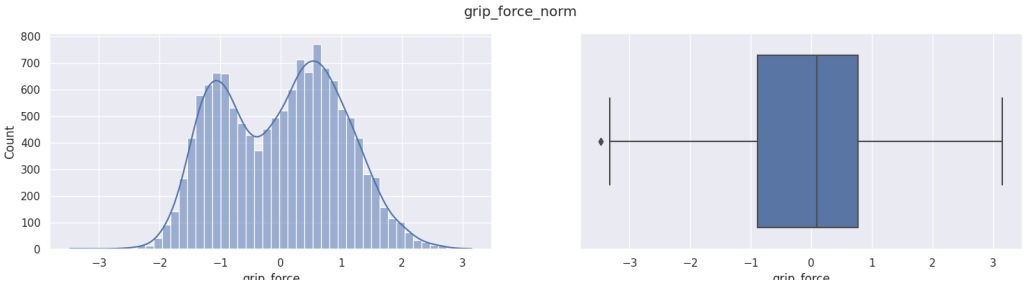


(b) Distribución posterior a normalizar.

Figura 1: Distribución de los datos del atributo *age*.



(a) Distribución previa a normalizar.



(b) Distribución posterior a normalizar.

Figura 2: GripNorm de los datos del atributo *grip_force*.

5.2. Análisis de componentes principales

Relacionado a la extracción de los atributos más relevantes, se logró determinar que dichos atributos fueron *age*, *gender*, *height_cm* y *weight_kg*, sumando en conjunuto una ganancia total de 85,36 %

Para la generación de un subespacio de componentes principales. Se obtuvieron 2 componentes que fueron utilizados de forma satisfactoria en etapas posteriores del desarrollo de este trabajo.

5.3. Pruebas simples de métodos de clustering

Observando la distribución real de los clusters del conjunto de datos utilizado (Figura 3), se puede determinar de forma visual que no existe un patrón claro en la distribución de los clusters, lo cual se debería tomar en cuenta al analizar los resultados de los demás métodos de clustering.

El método *K-means* (Figura: 4 logró obtener los 4 clusters solicitados, sin embargo, se notan discrepancias en la distribución real de los clusters y la propuesta por dicho algoritmo.

Por otro lado, el método *affinity propagation* (Figura: 5 mostró una alerta en su proceso de entrenamiento, donde se indica que los datos no están convergiendo, lo cual ocasiona que el algoritmo termine con poco más de 850 clusters en su totalidad.

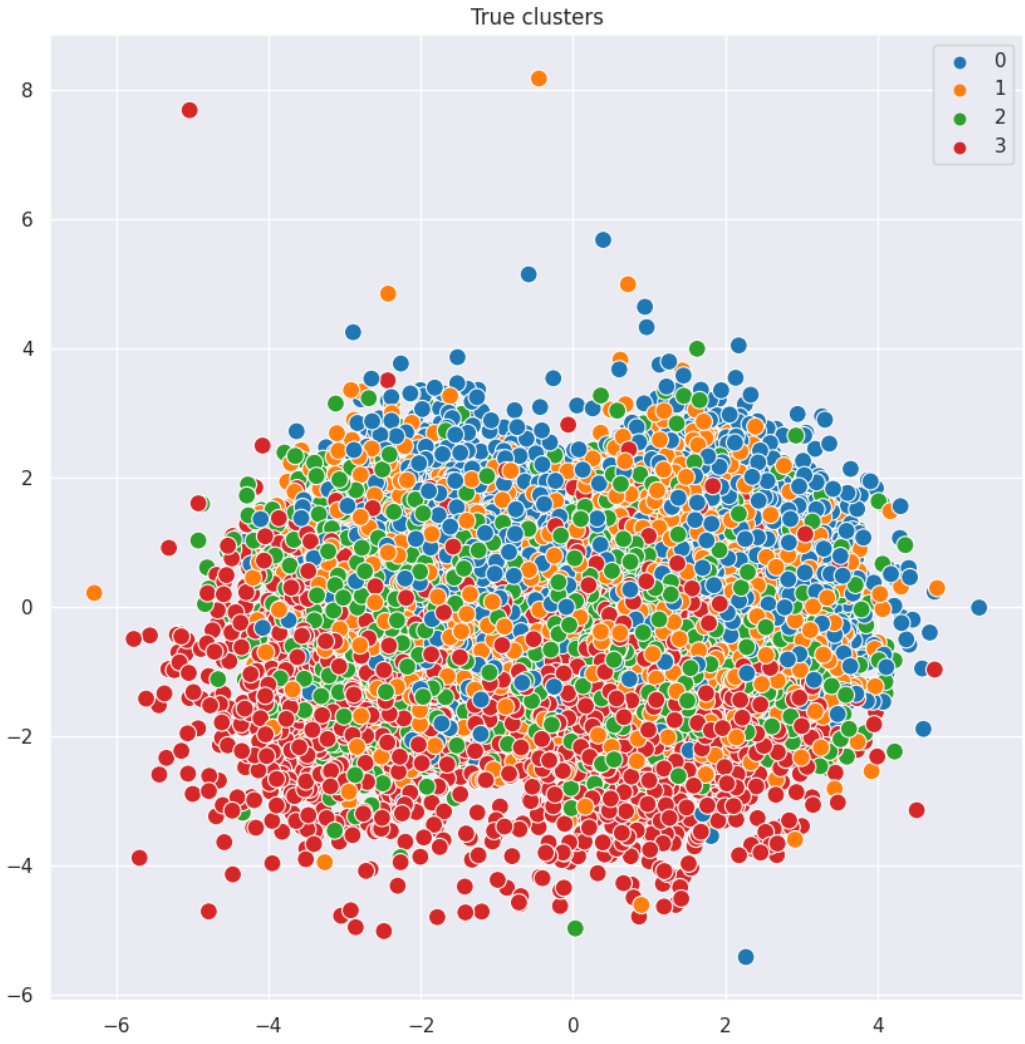


Figura 3: Distribución real de los clusters existentes en el conjunto de datos.

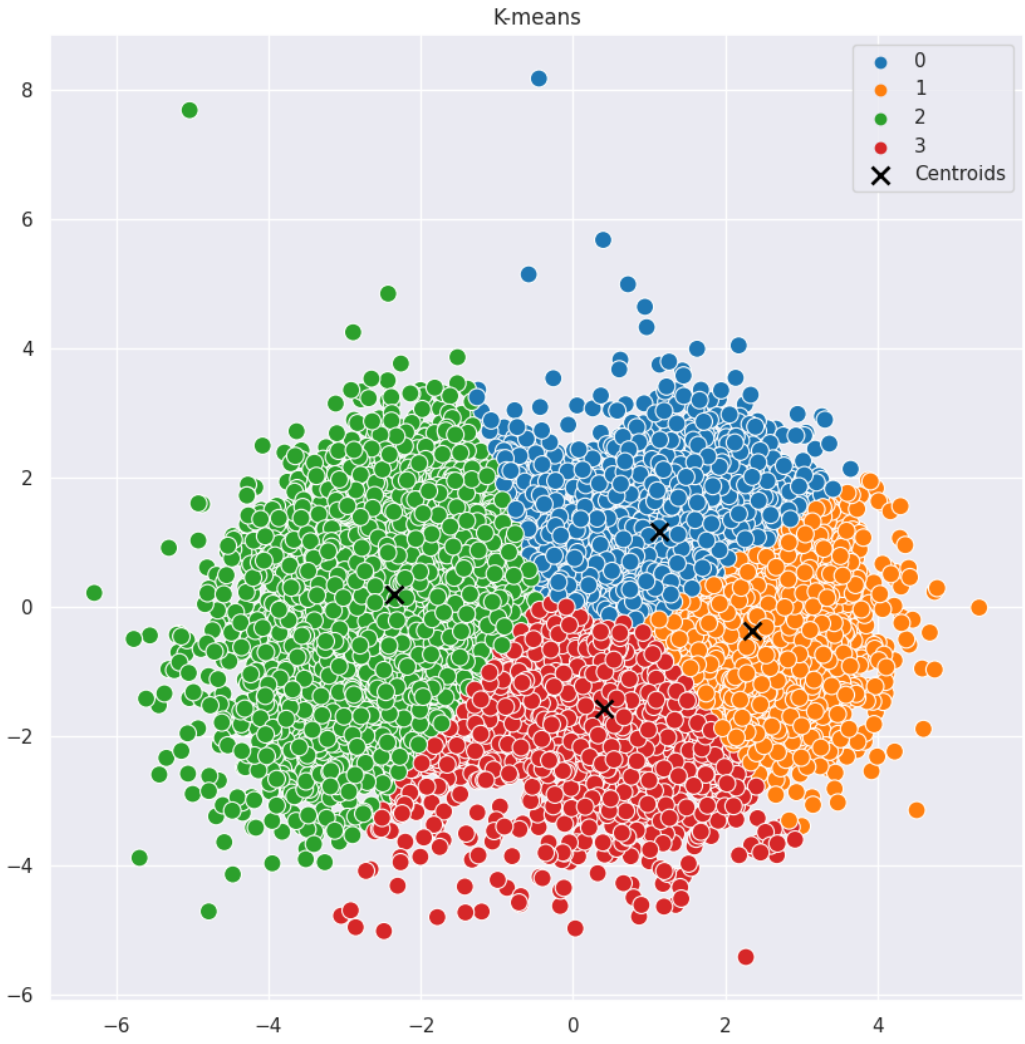


Figura 4: Distribución de los clusters propuestos por el algoritmo K-Means.

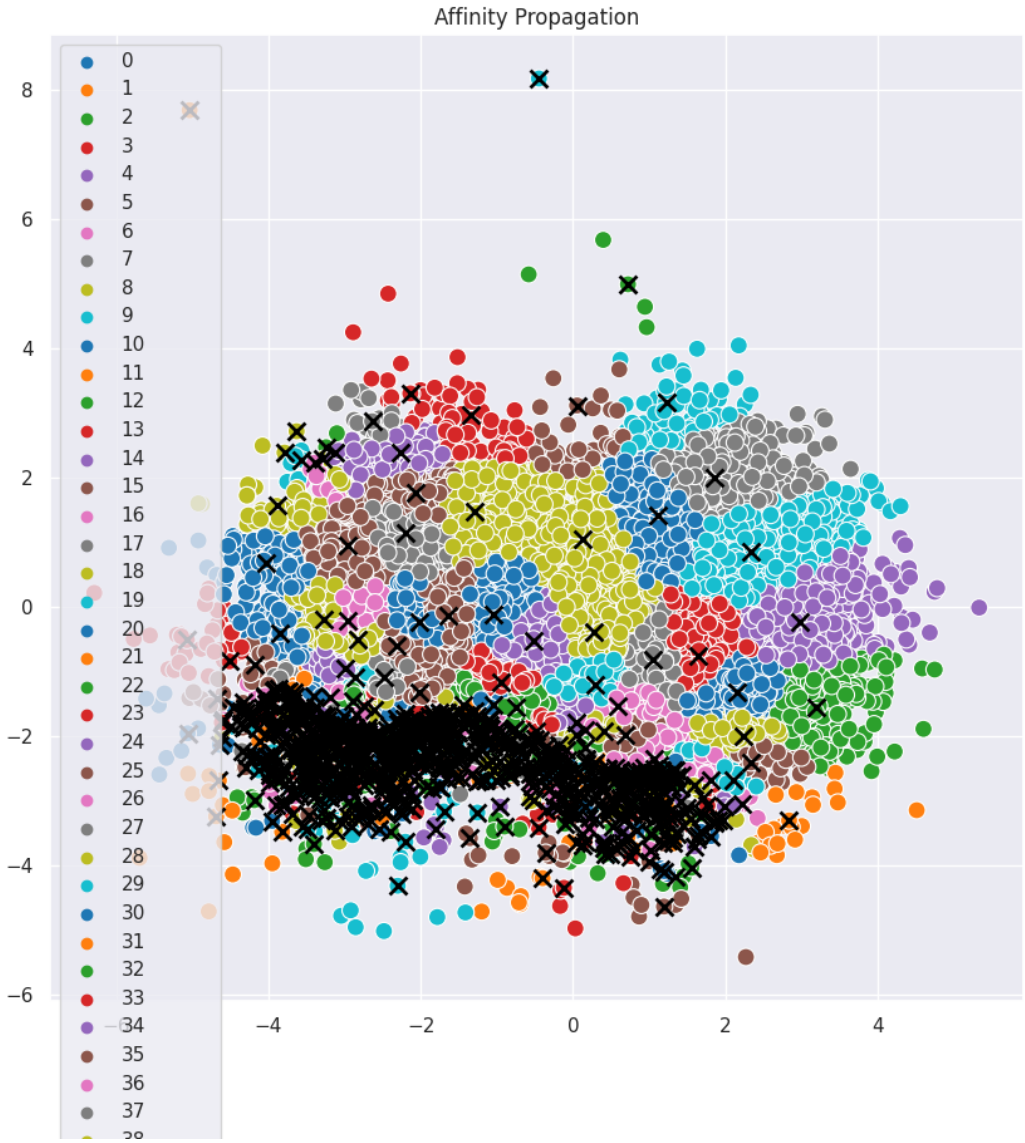


Figura 5: Distribución de los clusters propuestos por el algoritmo affinity.

5.4. Evaluación final de métodos de clustering

Debido a las grandes diferencias entre la cantidad de clusters determinada por el algoritmo *Affinity*, se optó por solamente evaluar el algoritmo *K-means* mendiante las pruebas de validación cruzada. Tomando un total de 10 repeticiones de entrenamiento se obtuvieron los resultados mostrados en la Figura 6.

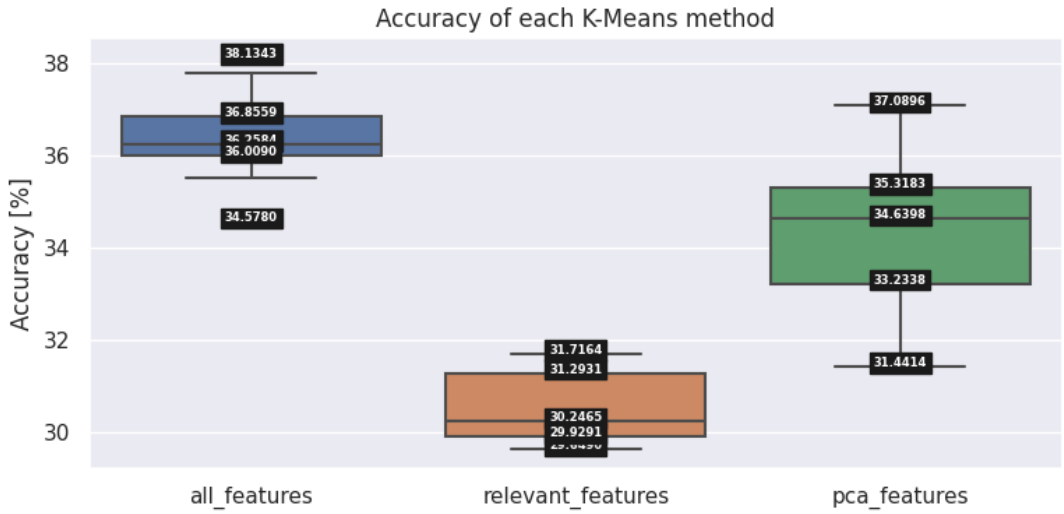


Figura 6: Exactitud resultante de la validación cruzada aplicada al método K-Means usando diferentes atributos.

6. Conclusiones

En este programa generalizado inicialmente se generaron funciones para preparar la base de datos en cuestión a fin de poder implementar posteriormente dos métodos de agrupamiento, esto con el propósito de identificar los puntos fuertes de cada uno de ellos.

Uno de los fuertes del algoritmo de k-means radica en su fácil interpretación, así como en su flexibilidad en el poder seleccionar la cantidad de clusters que se desea obtener. Este último punto es un deficit en el algoritmo affinity aplicado al conjunto de datos utilizado en este trabajo, ya que la misma naturaleza de los datos lo vuelve dificil de converger.

Es importante destacar la importancia de comprender los datos antes de adoptar alguna técnica de agrupamiento que se adapte bien a un conjunto de datos determinado para el problema en cuestión, ya que si se hace un análisis con respecto al algoritmo a utilizar puede brindar un ahorro el tiempo, así como la obtención de resultados más precisos.

Relacionado a las técnicas de reducción de datos, parecierá que resultaron tan efectivas para el caso de nuestro conjunto de datos, sin embargo, cabe mencionar que durante el proces de evaluación se pudo observar que el utlizar el subespacio generado por PCA generalmente permite que el algoritmo k-means alcance el punto de convergencia de forma más rápida que sin esos datos. Siendo específicos, sin usar PCA tardaba de 30 a 100 iteraciones en converger, mientras que con PCA se mantuvo por debajo de las 25 iteraciones.

Referencias bibliográficas

[1] M. A. Aceves Fernández, *Inteligencia artificial para programadores con prisa*. Universo de Letras, 2021, ISBN: 9788418854613.

[2] FreeCodeCamp, "8 algoritmos de agrupación en clústeres en el aprendizaje automático que todos los científicos de datos deben conocer," https://www.freecodecamp.org/espanol/news/8-algoritmos-de-agrupacion-en-clusteres-en-el-aprendizaje-automatico-que-todos-los-cientificos-de-dat n.d.

[3] A. IA, "Algoritmos de agrupamiento - aprende ia," https://aprendeia.com/algoritmos-de-clustering-agrupamiento-aprendizaje-no-supervisado/, n.d.

[4] L. Laura-Ochoa, "Evaluation of classification algorithms using cross validation," in *Ind. Innov. Infrastruct. Sustain. Cities Communities*, 2019, pp. 24–26.

A. Código documentado

El código completo y funcional se puede encontrar anexo en el archivo *zip* compartido en conjunto con este reporte.