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

Nueva versión del algoritmo de clúster gravitacional para la detección de datos atípicos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Alfonso Indurain Ibero

Javier Fernández Fernández

Pamplona

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

En este trabajo de fin de grado proponemos una modificación del algoritmo de clústering gravitacional para su utilización como detector de instancias atípicas. Comenzamos modificando la fórmula de la ley de gravitación universal. En lugar de utilizar la multiplicación para calcular la interacción entre las masas de las partículas, proponemos emplear una función más general para su cálculo. Del mismo modo, hemos estudiado la sustitución del sumatorio presente en esta misma fórmula utilizando en su lugar diferentes funciones de agregación. Finalmente, hemos incluido el algoritmo mencionado dentro de un sistema de toma de decisión, de modo que cada rama del sistema utilice conjuntos de datos diferentes. Con estas modificaciones estudiaremos su efectividad para la detección de instancias atípicas y compararemos los resultados de nuestra propuesta con aquellos obtenidos utilizando otros métodos clásicos de detección de anomalías, tanto supervisados como no supervisados.

Palabras clave

Algoritmo de clústering gravitacional, detección de outliers, funciones de agregación, esquema de toma de decisión, toma de decisión.

índice

1. Introducción 5

2. Preliminares 6

2.1. Algoritmo de clústering gravitacional 6

2.2. Funciones H 7

2.3. Funciones de agregación 7

2.4. Integrales difusas 7

2.5. Medidas de rendimiento 8

3. Algoritmo de clústering generalizado para detección de anomalías 9

3.1. Modificación del producto de masas 9

3.2. Generalización de la fuerza gravitacional 11

3.3. Esquema de toma de decisión 11

3.4. Búsqueda de partículas anómalos 12

3.5. Formulación del algoritmo 12

4. Experimentación 13

4.1. Datasets empleados en la experimentación 13

4.2. Optimización del algoritmo 14

4.3. Comparaciones 25

5. Conclusiones 28

6. Referencias 29

7. Anexo 31

7.1. Lenguaje de programación 31

7.2. Entorno de trabajo 31

7.3. Librerías 31

1. Introducción

Desde hace años en nuestro mundo se almacena toda la información posible. Ya sea almacenada en un pequeño servidor local o en el más potente de los servidores de Google, todos estos datos guardan un fin común, la extracción de conocimiento [1]. Sin embargo, debido a las inmensas cantidades de datos almacenados, su procesamiento y análisis sobrepasa las capacidades humanas, por lo que es común, en cierta medida, encontrar datos anómalos [2].

Un dato se considera anómalo cuando es muy diferente del resto de los elementos pertenecientes al mismo conjunto que él [3]. La presencia de estos datos anómalos puede entorpecer y modificar los resultados obtenidos tras aplicar metodologías de extracción de conocimiento, o por el contrario, estos datos pueden tratarse precisamente de las instancias que queremos encontrar [4],[5]. Es por ambos motivos que la búsqueda de instancias anómalas es un campo de gran importancia dentro de la minería de datos[6].

Las metodologías y algoritmos de minería de datos actuales pueden dividirse en dos grandes grupos claramente diferenciados, los métodos supervisados y los no supervisados [7], [8]. Los métodos supervisados basan su funcionamiento en la existencia de un conjunto de instancias clasificadas previamente [9]. Por el contrario, los métodos no supervisados no requieren de esta clasificación previa, basando su funcionamiento en las propiedades y características de los datos [10]. Ambas metodologías contienen diferentes algoritmos para la detección de estas instancias.

A pesar de la existencia de estos dos grupos de métodos, cuando nos enfrentamos a problemas del mundo real, no siempre existe la posibilidad de disponer de un conjunto de datos etiquetados. De hecho, es más común no disponer de esta información, por lo que se debe recurrir a los métodos de clasificación no supervisada. Para aquellos conjuntos de datos de los que no se dispone una clasificación previa, se han utilizado diferentes enfoques basados en algoritmos de agrupamiento [11] al igual que otros basados en la búsqueda de vecinos más cercanos [12]

Clústering en uno de los métodos de clasificación no supervisado más comunes y usados a la hora del tratamiento de datos [13]. Consiste en agrupar los datos en diferentes grupos de modo que los miembros de cada grupo sean lo más parecidos entre sí y diferentes los elementos de los demás grupos. Cada uno de estos grupos recibe el nombre de clúster. Uno de los métodos más comunes dentro de esta categoría es el algoritmo de clústering gravitacional[14].

Publicado en 1977, su autor W. E. Wright propuso crear un algoritmo de clústering basado en la ley de gravitación universal. El algoritmo de clústering gravitacional [14] utiliza los datos como partículas en un espacio Euclidiano, usando la fuerza gravitacional para atraerlas entre sí. De este modo las partículas irán uniéndose por el espacio formando clústeres de características similares. En la actualidad se han escrito diferentes artículos sobre el tema en cuestión basados en el movimiento de las partículas [15] [16].

Hoy en día los algoritmos de clasificación de outliers que tienden a obtener mejores clasificaciones se tratan de algoritmos supervisados. Esta clase de algoritmos disponen de una clara ventaja al basar su funcionamiento en un conjunto de instancias previamente clasificadas, sin embargo, esa misma característica también es su mayor inconveniente, ya que si no se dispone de subconjunto de instancias necesario no es posible aplicar este tipo de algoritmo. Con el objetivo de evitar este inconveniente, el algoritmo que presentamos en este trabajo se trata de un método no supervisado, el cual no necesita de un conjunto previamente clasificado y por tanto puede ser usado frente a cualquier conjunto de datos que se le presente.

El resto del trabajo se organiza de la siguiente manera. Hablaremos de los conocimientos necesarios para comprender este trabajo en la sección 2. En la sección 3 presentaremos el algoritmo propuesto. La experimentación será presentada en la sección 4, mientras que las conclusiones serán expuestas en la sección 5.

2. Preliminares

En esta primera sección explicaremos algunos de los fundamentos requeridos para la comprensión de este trabajo de fin de grado. Recordaremos las nociones de integrales Sugeno y Choquet, y la formulación del algoritmo de clúster original [11].

2.1. Algoritmo de clústering gravitacional

El algoritmo de clústering gravitacional es un algoritmo de clústering que forma clústeres simulando el movimiento de las partículas, calculado mediante la ley de gravitación universal de Newton.

$$F_i = G * \sum_{j \in \mathrm{N}, j \neq \mathrm{i}} \frac{m_i * m_j}{r_{i,j}} * \vec{u}_r$$

Al comienzo cada partícula representa un clúster. Inicialmente se dispondrá de tantos clústeres como partículas originales. A lo largo de su vida, las partículas se mueven afectadas por la presencia de las demás, uniéndose de forma que el número de clústeres va disminuyendo. Cuanto más tiempo este vivo un mismo clúster, más fuerte será la similitud entre sus miembros en comparación con el resto de los clústering. Como resultado final, se toma aquel estado con mayor porcentaje de vida total. Este estado representa la agrupación más estable de las partículas.

Pongamos un ejemplo, supongamos que tenemos un sistema con 200 partículas. Al comienzo de, las fusiones tienen lugar rápidamente resultando en 3 únicos clústeres restantes en t = 20, los cuales se encuentran separados ampliamente. Ahora supongamos que no es hasta el tiempo t=70 cuando dos de estos clústeres finalmente se unen, resultando en dos clústeres restantes, los cuales se fusionarán en t=80. Este algoritmo indicaría que el clústering con la mayor fuerza es con 3 clústeres distintos. Esto se debe a que es un estado muy fuerte debido a que ha estado vivo (70 - 20) / 80 *100% = 62.5% de la vida total del sistema.

Una de las características más significativas del algoritmo de clústering gravitacional es su independencia frente al número de clústeres. El algoritmo es capaz de realizar la clasificación de los datos sin la necesidad de indicarle el número de clases existentes.

Proceso del algoritmo:

1. Se especifica un número finito n de elementos o partículas $p_1, \dots, p_n$ mediante su posición o medidas $s_1, \dots, s_n$ y por sus masas $m_1, \dots, m_n$. Se asume que todas las medidas han sido propiamente escaladas.
2. Se declaran dos parámetros de aproximación δ y ∈. En cada intervalo de tiempo, el valor de dt es calculado de forma que la partícula más rápida se mueva una distancia δ. Cuando dos partículas están a una distancia ∈ o menor, se fusionan en una única partícula ubicada en su centroide. Un buen valor de ∈ = 2 * δ.
3. Se inicializa el tiempo t a 0. Los pasos d, e, f se repiten hasta que únicamente queda una partícula.

4. El movimiento de cada partícula restante i durante un intervalo de tiempo [t,t + dt] se calcula de acuerdo con la función g(i,t,dt), que es una función de los atributos de la partícula i y todas las partículas restantes.
5. La nueva posición $s_i(t+dt)$ de cada partícula restante i, se calcula como la suma $s_i(t+dt) = s_i(t) + g(i,t,dt)$. El tiempo t es incrementado en dt.
6. Si cualquier par de partículas i y j se han movido a una distancia ∈ entre ellas, entonces la partícula j se añade a la partícula i (asumimos i < j). Concretamente,

$$s_i(t) = \frac{\left(m_i(t) * s_i(t) + m_j(t) * s_j(t)\right)}{(m_i(t) + m_j(t))}, m_i(t) = (m_i(t) + m_j(t))$$

y la partícula j es eliminada.

7. Por cada estado por los que ha pasado el sistema, se selecciona el más estable y los clústeres finales son los presentes en este estado.

La función *g* utilizada en el paso d., es la siguiente:

$$g(i,t,dt) = dt^2 \sum_{j \in \mathrm{N(t)}, \mathrm{j} \neq \mathrm{i}} \frac{1}{m_i(t)} \frac{s_i(t) - s_j(t)}{\left|s_i(t) - s_j(t)\right|^3}$$

Esta función es conocida como la versión Markovian y esta explicada en mayor detalle en [11].

2.2. Funciones H

Una función $H: [0,1]^2 -> [0,1]$ es una función H si cumple las siguientes propiedades:

1. Es una función creciente.
2. Es una función simétrica.
3. Esta acotada en {1, … ,N} x {1, … , N}

2.3. Funciones de agregación

Una función de agregación en $\mathbb{R}^n$ es una función $A^{(n)}: \mathbb{R}^n \to [0,1]$ si cumple las siguientes propiedades:

- Es estrictamente creciente.
- $A(X) = 0 \; si \; y \; solo \; si \; \bigvee_{i=0}^{n} X_i = 0$
- $A(X) = 1 \; si \; y \; solo \; si \; \bigvee_{i=0}^{n} X_i = 1$

Durante esta memoria utilizaremos algunas funciones de agregación comunes junto con otras que no lo son tanto como la integral Choquet o la integral de Sugeno Generalizada que se han explicado previamente.

2.4. Integrales difusas

Supongamos que (X, F, μ) es un espacio medible, una medida difusa es una función μ: F →[0,∞] que cumple las siguientes propiedades[17]:

1. μ(ϕ) = 0
2. Si A, B ∈ F and A ⊆ B entonces μ (A) ≤ μ (B)
3. Si $A_n \in F \; and \; A_1 \subseteq A_2 \subseteq A_3 \subseteq A_4 \ldots . then \; \lim_{n\to\infty} \mu(A_n) = \mu(\lim_{n\to\infty}(A_n))$

Un ejemplo comúnmente usado de mediad difusa es: siendo “m” la medida sobre el conjunto A:

$$m = 1 / |A|$$

Finalmente, una integral difusa es una integral respecto a una medida difusa.[17]

2.4.1. Integral de Sugeno

En el artículo [17], Sugeno define la conocida como integral de Sugeno.

Supongamos que μ es una medida difusa sobre X, la integral de Sugeno de una función f : X→[0, 1] se define como:

$$\int f(x) d\mu = \max_{1 \le i \le n} (\min (f(x_i), \mu(A_i)))$$

Donde $\{ f(x_1), f(x_2), \dots, f(x_n)\}$ son rangos y están definidos como

$$\{ f(x_1) \le f(x_2) \le \cdots \le f(x_n)\}$$

En este trabajo, emplearemos una modificación de la integral de Sugeno con la siguiente forma:

$$f(x) = \sum_{i=1}^{n} sum(\text{prod } (f(x_i), \mu(A_i)))$$

2.4.2. Integral Choquet

Siendo $\mu : 2^{[n]} \to [0,1]$ una medida difusa. La integral Choquet con respecto a la medida difusa μ viene dada por:

$$Ch_{\mu}(X) = \sum_{i=1}^{n} \left(x_{(i)} - x_{(i-1)}\right) \cdot \mu\,(E_{(i)})$$

Para cualquier $x = (x_1, \dots, x_n) \in [0, \infty]^n$, donde (·) es una permutación en [n] tal que $x_{(1)} \le \cdots \le x_{(n)}$, por convenio $x_{(0)} = 0$ y $E_{(i)} = \{(i), \dots, (n)\}$ para $i = 1, \dots, n$.

2.5. Medidas de rendimiento

Existen diferentes medidas de rendimiento, una de las más comunes es la precisión. La precisión se usa para medir el número de aciertos de la clase positiva. En el caso de los outliers, esta medida no aporta gran información, incluso puede llegar a aportar información errónea.

Supongamos que tenemos un problema de clasificación de outliers. Nuestro dataset cuenta con 1000 ejemplos de los cuales únicamente 10 son outliers. Si creamos un modelo que clasifica todos los ejemplos como no outliers, estaríamos obteniendo una precisión de 990/1000, es decir del 99%. Sin embargo, esta es una medida errónea ya que no hemos sido capaces de clasificar correctamente ningún outlier.

Por este motivo, es común usar una matriz de confusión como la siguiente

	Realmente es positivo	Realmente es negativo
Predicho como positivo	Verdaderos Positivos (VP)	Falsos Positivos (FP)
Predicho como negativo	Falsos Negativos (FN)	Verdaderos Negativos (VN)

Ilustración 1 Matriz de confusión

Siendo:

- VP: Número de ejemplos de la clase positiva clasificados correctamente.
- FP: Número de ejemplos de la clase negativa clasificados erróneamente.
- FN: Número de ejemplos de la clase positiva clasificados erróneamente.
- VN: Número de ejemplos de la clase negativa clasificados correctamente.

Con esta tabla se pueden sacar diferentes medidas de rendimiento como:

- Precisión: $\frac{VP}{VP+FP}$, de todos los ejemplos clasificados como clase positiva, cuantos lo son realmente.
- Recall: $\frac{VP}{VP+FN}$, de todos los ejemplos de la clase positiva, cuantos se han clasificado correctamente.
- F1-Score: $2 * \frac{Precisión*Recall}{Precisión+Recall}$, se trata de una medida armónica que mide el balance entre la precisión y el recall.

3. Algoritmo de clústering generalizado para detección de anomalías

En este capítulo explicaremos el algoritmo desarrollado en este trabajo de fin de grado, diferenciando tres posibles modificaciones sobre el mismo.

3.1. Modificación del producto de masas

Originalmente, en el artículo [14] se utiliza la siguiente función g(i, t, dt):

$$g(i,t,dt) = \sum \left(\frac{m_i * m_j}{m_i} * \frac{s_j - s_i}{|s_j - s_i|^3} \right)$$

Para este trabajo, proponemos sustituir el producto de las masas mediante una función H de la siguiente forma:

$$g(i,t,dt) = \sum \left(\frac{H(m_i, m_j)}{m_i} * \frac{s_j - s_i}{|s_j - s_i|^3} \right)$$

Teniendo:

$$H_o(x,y)\begin{cases} B_1 & si\ G_S(x,y) < B_1 \\ B_2 & si\ G_S(x,y) > B_2 \\ G_S(x,y) & si\ B_1 \leq G_S(x,y) \leq B_2 \end{cases}$$

con $0 < B_1 < B_2$.

Donde $G_S(x,y)$ se trata de cualquier función de overlap.

Una función $O: [0,1]^n -> [0,1] con\ n \geq 2$ es una función de overlap si cumple las siguientes propiedades:

1. O es simétrica
2. $O(X_1, \dots, X_n) = 0$ si y solo si $\prod_{i=1}^{n} X_i = 0$
3. $O(X_1, \dots, X_n) = 1$ si y solo si $\prod_{i=1}^{n} X_i = 1$
4. O es creciente
5. O es continua en cada variable

En este trabajo, hemos tomado como función de overlap

$$G_S(x,y) = (x * y)^c$$

donde c será una de nuestras variables de estudio. En este punto es posible sustituir la función de overlap por otra diferente a la utilizada, dejando como posible estudio el uso de diferentes funciones de overlap para un estudio futuro.

Los parámetros B1 y B2 se usan para acelerar o decelerar el proceso de clústering. B1 se utiliza para acelerar el proceso al ser utilizado como límite inferior de la función H. Las partículas más pequeñas irán más rápido debido a que todos los valores tendrán un valor mínimo igual a B1, mientras que en la multiplicación este valor mínimo es 0. Por el lado contrario, B2 se utiliza como límite superior, limitando la influencia y velocidad de las partículas mayores.

Se presentan dos posibles modelos, uno estático y otro dinámico.

3.1.1. Versión estática de la función H

En el modelo estático, los valores de B1 y B2 deben indicarse desde el comienzo y no varían su valor en ningún instante, requiriendo un estudio para la obtención de valores óptimos.

La presencia de outliers puede hacer que sea difícil o incluso imposible la obtención de unos valores óptimos para B1 y B2, debido a que pueden crear estados más estables debido a su lejanía con el resto de los datos. En este caso, el modelo dinámico es ofrecido para intentar solventar este problema.

3.1.2. Versión dinámica de la función H

El modelo dinámico recalcula el valor de B1 y B2 en cada iteración, con el fin de acelerar el movimiento de las partículas outlier y así conseguir configuraciones potencialmente óptimas, más estables. En este caso, B1 y B2 se pueden inicializar a 0 y 1 respectivamente. La actualización de sus valores se realiza siguiendo el siguiente algoritmo:

$aug = 0.1, dec = 0.1$
  $if \max(masses) \geq K\ sum(masses) then$
    $B_1 = \min(B_1 + aug, B_2)$
    $if\ B_1 = B_2\ then$
      $B_2 = B_2 + aug$
    $endif$
  $else$
    $B_2 = \max(B_2 - dec, B_1)$
    $if\ B_1 = B_2\ then$
      $B_1 = \max(0, B_1 - dec)$
    $endif$
  $endif$

3.2. Generalización de la fuerza gravitacional

En el artículo original[14], el autor utiliza la siguiente fórmula para el cálculo del diferencial de tiempo empleado para determinar la duración de cada iteración:

$$dt = \sqrt{\frac{\delta}{\frac{1}{m_i} * \sum_{j=0}^{n} m_i * m_j * \frac{s_j - s_i}{|s_j - s_i|^3}}}$$

En este trabajo sugerimos sustituir la formula previa por la siguiente

$$dt = \sqrt{\frac{\delta}{\frac{1}{m_i} * agg(H(m_i, m_j) * \frac{s_j - s_i}{|s_j - s_i|^3})}}$$

De igual modo, la función g(i,t,dt) original, se verá modificada del mismo modo:

$$g(i,t,dt) = \sum_{j=0}^{n} \frac{m_i * m_j}{m_i} * \frac{s_j - s_i}{|s_j - s_i|^3}$$

pasa a ser:

$$g(i,t,dt) = agg\left(\frac{H(m_i, m_j)}{m_i} * \frac{s_j - s_i}{|s_j - s_i|^3}\right)$$

3.3. Esquema de toma de decisión

Con las modificaciones propuestas en las dos secciones previas, vamos a hacer uso de un esquema de toma de decisión para mejorar el conocimiento extraído por cada algoritmo.

Este esquema de toma de decisión está compuesto únicamente por diferentes ejecuciones de nuestro algoritmo gravitacional, donde cada ejecución se diferencia del resto según el

subconjunto de instancias del dataset y los atributos empleados de las instancias, que recibe como entrada.

3.4. Búsqueda de partículas anómalos

Para la selección de las posibles instancias anómalas, cada algoritmo gravitacional aplica las siguientes reglas:

- Se seleccionan aquellos estados por los que haya pasado el algoritmo que tengan un tiempo de vida mínimo T_i , donde:

$$T_i \geq pt * T, \qquad \forall\, 1 < i < N$$

La variable *pt* es una de las variables de estudio dentro de este trabajo.

- De aquellos estados seleccionados en el paso anterior, se seleccionan los clusters que tengan un tamaño máximo Np_i, siendo el tamaño de un cluster viene representado por el número de partículas que lo conforman:

$$Np_i \leq p * N$$

La variable *p* es una de las variables de estudio dentro de este trabajo y *N* es el número total de partículas en el sistema.

- Finalmente, una vez obtenidas las partículas seleccionadas por cada algoritmo gravitacional, se asigna un peso a todas las partículas del sistema. El peso de la partícula X se calcula como:

$$P(X) = \frac{número\ de\ gravitacionales\ que\ clasifican\ la\ particula\ X\ como\ anómala}{número\ total\ de\ gravitacionales}$$

Una vez cada partícula tiene un peso asignado, se agregan los diferentes pesos utilizando una función de agregación para obtener una cota.

$$cota = Agg(P_1, P_2, P_3, \ldots, P_n), siendo\ n\ el\ número\ total\ de\ partículas$$

Finalmente se seleccionan como instancias anómalas todas aquellas cuyo peso sea superior al valor de la cota.

3.5. Formulación del algoritmo

Con todas las modificaciones previas, el algoritmo propuesto queda definido de la siguiente manera:

1. Se selecciona el valor de las diferentes variables: *k*, *c*, *p*, *pt*, *numAtr*, *numArb*, *percTest*, *agg1* y *agg2*.
2. Se dividen los datos de entrada en dos conjuntos de train_1 y test_1.
3. Los pasos 4, 5, 6, 7, 8 se repiten numArb veces.
4. Se divide el conjunto train_1 en dos nuevos subconjuntos train_2 y test_2.
5. Se selecciona únicamente el *numAtr* porciento de los atributos de train_2.
6. Utilizando la nueva función g(i,t,dt) y el nuevo método de cálculo de *dt,* previamente explicados, se ejecuta el algoritmo gravitacional para el conjunto train_2.
7. Se seleccionan las instancias anómalas según la sección 3.4

El algoritmo presentado cuenta con 3 versiones diferentes. La primera se trata de la versión básica, donde usaremos la función Markovian original. La segunda versión es la estática, la cual utilizará una función H, con valores de B1 y B2 constantes, para sustituir al producto de las masas. La última versión es la dinámica , la cual utilizará una función H, con valores de B1 y B2 dinámicos, para sustituir al producto de las masas. Todas ellas serán incluidas en un esquema de toma de decisión y sustituirán el sumatorio de la fórmula de atracción gravitacional, por diferentes funciones de agregación.

4. Experimentación

En esta sección del trabajo presentaremos los dataset usados para las diferentes pruebas. A continuación realizaremos un ajuste de los parámetros del algoritmo, seguido de una selección de la versión óptima del problema. Finalizaremos optimizando los parámetros del esquema de toma de decisión y realizando una comparación con los resultados obtenidos frente a otros algoritmos actuales.

4.1. Datasets empleados en la experimentación

Para realizar la optimización de los parámetros del algoritmo, hemos empleado 3 dataset.

Dataset	Número de clases	Número Atributos	Número Instancias
Ecoli	8	7	336
Wisconsin	2	9	683
Cardio	3	21	1831

Tabla 1 Dataset para la optimización de parámetros

Los dos primeros son Ecoli y Wisconsin, los cuales hemos tomado de la página KEEL[20].

El dataset Ecoli cuenta con un total de 8 clases. Hemos agrupado estas 8 clases en 2 grupos. El primero formado por las clases '*imS*', '*imL*' y '*omL*' representan las instancias outliers al tratarse de clases minoritarias. El segundo grupo está formado por todas las demás clases y representan los datos no atípicos. El dataset Wisconsin contiene 2 clases, siendo la clase positiva la minoritaria, por lo que será la utilizada para representar las instancias atípicas.

El último de estos tres dataset, Cardio, se trata del mismo empleado en el artículo [24]. Este dataset contiene información de ratios de latidos en fetos, los cuales están agrupados en 3 clases: '*normal*', '*suspect*' y 'pathologic'. De estas tres clases hemos descartado la clase '*suspect*' al igual que en el artículo y hemos empleado la clase 'pathologic' para representar los datos atípicos.

Para la realización de las comparaciones y optimización de los parámetros del esquema de toma de decisión, hemos selección un total de 5 datasets, todos ellos obtenidos de la página KEEL [20].

La sección de datasets desbalanceados de KEEL contiene datasets con dos clases, una de ellas ampliamente mayoritaria frente a la otra. Dada esta característica de estos conjuntos de datos, podemos hacer una equiparación donde:

- La clase mayoritaria(negativa) representa el conjunto de datos inlier, lo que en otro dataset entenderíamos como datos normales.
- La clase minoritaria(positiva) representa el conjunto de datos outlier, aquellos que nos interesa detectar y clasificar correctamente.

KEEL cuenta con un total de 145 datasets con estas características. Cada uno de ellos posee una clasificación llamada IR que mide el ratio de desbalanceo del dataset.

El IR representa el ratio de desbalance de un dataset y es calculado como el porcentaje de datos de la clase mayoritaria frente al porcentaje de datos de la clase minoritaria.

Para el primer conjunto de pruebas, hemos seleccionado 5 datasets cuyos IR varían en el rango 1.5 – 9, los más bajos que ofrecen y por tanto siendo un buen punto de partida.

4.1.1. Datasets ratio 1.5 – 9

Dataset	Índice IR	Número Atributos	Número Instancias
Ecoli1	3.36	7	336
Ecoli3	8.6	7	336
Glass6	6.38	9	214
Wisconsin	1.86	9	683
Yeast3	8.1	8	1484

Tabla 2 Datasets para las comparaciones

Antes de poder utilizar estos dataset, hemos realizado un preprocesamiento de los datos mediante una z-score.

4.2. Optimización del algoritmo

En esta sección expondremos las pruebas realizadas para el ajuste de los diferentes parámetros usados en el algoritmo propuesto. Intentaremos obtener los mejores valores para cada uno de los parámetros de forma independiente con el fin de obtener la mejor configuración posible.

Finalizaremos la sección realizando un estudio sobre las diferentes versiones del algoritmo y realizando una optimización del esquema de toma de decisión tras seleccionar la versión del algoritmo con mejor rendimiento.

Comenzamos esta sección centrándonos en la optimización de los siguientes 4 parámetros:

Parámetro	Descripción
C	Parámetro utilizado para modificar la función de overlap utilizada en la función H
K	Parámetro utilizado para la optimización de la agregación de masas en la función H.
P	Umbral para el tamaño de los clusters seleccionados cuyas partículas se seleccionan como posibles instancias atípicas.
PT	Umbral para el tiempo de vida de los clusters seleccionados cuyas partículas se seleccionan como posibles instancias atípicas.

Tabla 3 Parámetros a optimizar

4.2.1. Optimización de la función de overlap

La variable *c* se utiliza como parte de la función de overlap utilizada en la función H

$$H_o(x,y) = \begin{cases} B_1 & si\ G_S(x,y) < B_1 \\ B_2 & si\ G_S(x,y) > B_2 \\ G_S(x,y) & si\ B_1 \leq G_S(x,y) \leq B_2 \end{cases}$$

donde $G_S(x,y)$ se trataba de una función de overlap de la forma

$$G_S(x, y) = (x * y)^c$$

El valor de c nos permitirá estudiar el comportamiento del detector de instancias atípicas frente a ligeras modificaciones en la función de overlap.

En las siguientes tablas se puede observar un resumen de todas las pruebas realizadas para la optimización de este parámetro:

Dataset	Ecoli	Cardio	Wisconsin
C = 0.5	0.34283996	0.21060449	0.78010534
C = 1	0.2951746	0.18948511	0.76752786
C = 2	0.312880356	0.20813065	0.77752327

Tabla 4 F1-Score para el método del voto

Dataset	Ecoli	Cardio	Wisconsin
C = 0.5	0.145684771	0.222401986	0.780105338
C = 1	0.10594216	0.17591885	0.76752786
C = 2	0.10482697	0.20076097	0.77752327

Tabla 5 F1-Score para las agregaciones

Tanto la tabla 4 como la tabla 5 muestran que, de los tres valores estudiados, el que mejor resultados aporta es claramente 0.5. En el dataset Wisconsin la diferencia en los resultados es escasa, sin embargo, cuando se tiene en cuenta los dataset Ecoli y Cardio los resultados son bastante más claros.

Finalmente, queda por resolver que es mejor utilizar, la toma de decisión mediante el método del voto, con su consecuente estudio de valores para el threshold, o utilizar una función de agregación que nos aporte un único threshold independiente del dataset utilizado.

Para llevar a cabo esta decisión, seguiremos mostrando los resultados utilizando ambas versiones durante la optimización de los tres parámetros restantes. Una vez realizadas las pruebas necesarias tomaremos una decisión acorde a los resultados obtenidos en cada una de ellas.

4.2.2. Optimización de la agregación de masas

Esta variable se emplea para controlar la actualización de las variables B1 y B2, sirviendo como threshold a la hora de decidir si el valor de B1 y B2 debe aumentar o disminuir.

K se trata del porcentaje mínimo que debe pesar la mayor partícula en cada estado del algoritmo para actualizar el valor de B1 y B2. Tras un primer estudio general, se pudo observar que el rango óptimo para este valor es [0.1 – 0.25]. Mas adelante, estudiaremos diferentes valores dentro de este rango para obtener un valor óptimo para nuestro problema.

Dataset	Ecoli	Cardio	Wisconsin
K = 0.1	0.234111898	0.204083036	0.772692596
K = 0.12	0.30091954	0.200578262	0.777665437
K = 0.14	0.235605426	0.215869633	0.772304005
K = 0.16	0.159950381	0.184379291	0.765363538
K = 0.18	0.187453249	0.242836387	0.769879533
K = 0.2	0.325505051	0.232104001	0.772876599

Tabla 6 F1-Score para el método del voto

Dataset	Ecoli	Cardio	Wisconsin
K = 0.1	0.1198747	0.171666427	0.772692596
K = 0.12	0.173809524	0.185999751	0.777665437
K = 0.14	0.127842841	0.195086333	0.772304005
K = 0.16	0.093506494	0.19239	0.765363538
K = 0.18	0.123129184	0.235989161	0.769879533
K = 0.2	0.13938934	0.212267283	0.772876599

Tabla 7 F1-Score para las agregaciones

Tras la realización de todas las pruebas anteriores, los resultados pueden simplificarse en las tablas 6 y 7.

La tabla 6 contiene la mejor puntuación obtenida para cada dataset con los diferentes valores de *k*. Cada uno de estos valores ha sido obtenido utilizando diferentes umbrales dependiendo del dataset y de la variable *k*. Observando los resultados de esta tabla, es fácil ver que las puntuaciones obtenidas para *k* = 0.12 y *k* = 0.2 destacan frente a las demás.

Los resultados obtenidos en la tabla 7 representan un resumen de las pruebas anteriores realizadas con las funciones de agregación como método de toma de decisión. Al igual que cuando utilizamos el método del voto, aparentemente se obtienen mejores resultados cuando se utilizan los mismos valores, sin embargo, en esta ocasión el valor *k* = 0.18 también obtiene buenos resultados.

Teniendo en cuenta ambas tablas, decidimos optar por 0.12 como valor óptimo para las pruebas futuras ya que ofrece un mejor rendimiento al utilizar funciones de agregación frente a 0.2, el cual funciona mejor utilizando el método de voto. Independientemente de esto, ambos valores han probado obtener los mejores resultados en el conjunto global de los datos y podría utilizarse cualquiera de los dos.

4.2.3. Optimización del tamaño de los clusters anómalos

La variable p es una de las dos variables empleadas directamente en la clasificación de una partícula como atípica. Una vez se dispone de todos los estados por los que ha pasado el algoritmo gravitacional, se seleccionan únicamente aquellos estados que hayan estado vivos al menos un porcentaje de tiempo igual a P:

$$Np_i \leq p * N$$

siendo N el número total de partículas del conjunto train_2 y Np_i el número de partículas del clúster *i*.

El estudio de diferentes valores de esta variable nos permitirá evaluar cuales son los estados más importantes de cara a la detección de instancias atípicas. Un valor muy pequeño en esta variable incluirá estados menos estables, mientras que aumentar el valor de esta permitirá seleccionar únicamente los estados cuyas configuraciones serían más optimas de cara a un proceso de clústering.

Independientemente de esto, si bien es cierto que las instancias atípicas tienden a crear configuraciones estables, esto no siempre es cierto y por tanto no podemos limitarnos únicamente a tomar los *n* estados más estables, motivo por el cual estudiaremos diferentes threshold para esta selección.

Debido a que esta variable se relaciona directamente con el tamaño de los clústeres, debido a la definición de instancia atípica, el valor de esta variable debe ser pequeño. Teniendo esto en cuenta, hemos realizado las siguientes pruebas.

Dataset	Ecoli	Cardio	Wisconsin
P = 0.01	0.0511096	0.1742752	0.77321415
P = 0.02	0.0754386	0.17298013	0.78043039
P = 0.05	0.19359095	0.25462568	0.77684872
P = 0.1	0.31853372	0.16406162	0.77407017
P = 0.2	0.36603896	0.18124419	0.7728959

Tabla 8 F1-Score para el método del voto

Dataset	Ecoli	Cardio	Wisconsin
P = 0.01	0.05054404	0.1742752	0.77321415
P = 0.02	0.08880826	0.17303227	0.78043039
P = 0.05	0.08444444	0.2254456	0.77684872
P = 0.1	0.1	0.16406162	0.77407017
P = 0.2	0.11857566	0.18213633	0.7728959

Tabla 9 F1-Score para las agregaciones

Las tablas 8 y 9 muestran un resumen de las pruebas realizadas previamente.

Comenzamos hablando sobre los resultados obtenidos en el dataset Wisconsin. Como hemos comentado anteriormente, variar el valor de la variable no parece surgir ningún efecto notorio sobre las puntuaciones obtenidas en este dataset, debido a esto utilizaremos únicamente los resultados obtenidos en los otros dos dataset para decidir un valor óptimo para la variable *p*.

El dataset Ecoli muestra una tendencia a mejorar los resultados conforme aumentamos el valor de la variable *p*, mientras que el dataset Cardio obtiene los mejores resultados al utilizar *p* = 0.05.

Debemos señalar que el dataset Cardio cuenta con un número superior de instancias que Ecoli. Concretamente, el dataset Cardio cuenta con un total de 1831 partículas mientras que el dataset Ecoli cuenta con únicamente 336. Haciendo los cálculos del tamaño máximo de partículas por clúster utilizado en cada dataset, ambos acaban siendo similares. Ecoli selecciona clústeres formados por 84 partículas o menores, mientras que Cardio selecciona clústeres de 91 partículas o menores.

Estos resultados no son suficientes para implantar un único valor óptimo para la variable *p*, pero si para indicar que, cuanto mayor sea el número de instancias del dataset, menor habrá de ser el valor de la variable y viceversa.

Recomendamos un estudio previo, acorde a lo previamente estipulado, para cada dataset utilizado. Aun así, utilizar 0.05 para dataset de mayor tamaño y 0.25 para los de menor tamaño es un buen punto de partida para esta variable.

4.2.4. Optimización del tiempo de vida de los clusters anómalos.

La variable pt es la segunda variable empleada directamente en la clasificación de una partícula como atípica. Tras filtrar los diferentes estados del algoritmo mediante la variable *p*, disponemos de un conjunto de clústeres, que incluyen a todas las partículas del sistema. De todos estos clústeres, únicamente seleccionamos aquellos cuyo tamaño cumple la siguiente condición:

$$T_i \geq pt * T, \qquad \forall\, 1 < i < N$$

siendo N el número de estados, T el tiempo de vida total del sistema y T_i el tiempo de vida del estado *i*

Utilizando diferentes valores para la variable *pt* se limita el número máximo de partículas que pueden pertenecer a un mismo clúster y seguir considerándose atípicas. Realizamos este estudio debido a que las instancias atípicas no siempre son partículas completamente independientes del resto, en ocasiones estos datos pueden venir en pequeños grupos que forman pequeños clústeres rápidamente.

Las siguientes tablas muestran el resumen de las pruebas realizadas para la optimización de este parámetro:

Dataset	Ecoli	Cardio	Wisconsin
PT = 0.01	0.1865252	0.18628226	0.5391312
PT = 0.05	0.26892705	0.19943123	0.75937058
PT = 0.1	0.19265635	0.25468957	0.78771176
PT = 0.2	0.27217756	0.21680937	0.7926302

Tabla 10 F1-Score para el método del voto

Dataset	Ecoli	Cardio	Wisconsin
PT = 0.01	0.1028707	0.17834516	0.5391312
PT = 0.05	0.12509566	0.20827139	0.75937058
PT = 0.1	0.11914231	0.26562443	0.78771176
PT = 0.2	0.0968938	0.22019804	0.7926302

Tabla 11 F1-Score para las agregaciones

Las tablas 10 y 11 muestran un resumen de todas las pruebas realizadas en esta sección. Si tenemos en cuenta únicamente los resultados obtenidos mediante el método del voto, pare bastante claro que el valor optimo a elegir sería 0.2, sin embargo, el rendimiento de este valor al utilizar funciones de agregación es claramente inferior.

Debido a esto, recomendamos utilizar 0.1 como valor óptimo para la variable *pt*, puede que su rendimiento no sea el mejor si nos fijamos en los resultados obtenidos en el dataset Ecoli, sin embargo, es el valor que mejor rendimiento ofrece si tenemos en cuenta ambos métodos de toma de decisión y todos los dataset. Dicho esto, también es posible encontrar valores que funcionen mejor frente a un dataset concreto, como es el caso al utilizar 0.05 y el dataset Ecoli.

4.2.5. Pruebas para las tres versiones del algoritmo

Tras haber seleccionado el conjunto de datasets que vamos a utilizar para este primer conjunto de pruebas, comenzaremos estudiando cuál de las tres configuraciones propuestas ofrece un mejor rendimiento a la hora de clasificar instancias outliers. Recordamos que las tres configuraciones propuestas son:

- Incluir el algoritmo gravitacional dentro de un esquema de toma de decisión y sustituir la función de agregación original(sumatorio) por otra diferente.
- A la propuesta anterior, añadir la versión estática de las funciones H.
- A la primera propuesta, añadir la versión dinámica de las funciones H.

Dado que la segunda propuesta puede utilizar un numero indefinido de configuraciones distintas para los valores iniciales de las variables B1 y B2, estudiaremos únicamente una pequeño subconjunto de todas estas posibles combinaciones.

4.2.5.1. Versión sin función H

Dentro del mundo de las funciones de agregación, existen cientos de funciones distintas. Es imposible saber cuáles de ellas ofrecerán un mejor rendimiento frente a otras, de igual modo, debido a las limitaciones de tiempo también es imposible probarlas todas y cada una de ellas para obtener la mejor.

Sin embargo, aun disponiendo del tiempo y recursos necesarios para implementar y testar todas las funciones de agregación existentes hasta el día de hoy, cada conjunto de datos es un mundo en sí mismo y por lo tanto agregaciones que funcionan bien en un dataset, pueden funcionar muy mal en otros. Debido a esto, hemos seleccionado un pequeño conjunto de funciones de agregación que hemos considerado las más comunes y que generalmente ofrecen mejores resultados en un conjunto global.

Las funciones de agregación que estudiaremos en lugar del sumatorio son: media, mediana, Choquet y Sugeno Generalizada.

Fijamos el conjunto de entrenamiento a 0.6 (60%),el porcentaje de atributos a 0.7 (70%), el número de árboles a 15, la variable delta a 0.03 y la variable épsilon a 0.06. Finalmente, ejecutaremos cada prueba un total de 5 veces y obtendremos el media de las 5 ejecuciones como resultado final de prueba.

Dataset\Agregación	Sumatorio	Media	Mediana	Choquet	Sugeno
Ecoli 1	0.33072891	0.45231236	0.86780968	0.419971981	0.85184436
Ecoli 3	0.44787016	0.43556336	0.9391281	0.43870329	0.94148509
Glass 6	0.30933602	0.348662253	0.88158132	0.4753112	0.39832235
Wisconsin	0.05903881	0.0547649	0.7877641	0.13131742	0.78115
Yeast 3	0.10187155	0.09499341	0.85321621	0.16142745	0.12027442

Tabla 12 F1-Score para las diferentes agregaciones

Comparamos los resultados utilizando diferentes funciones de agregación. Para comparar los resultados utilizamos la medida F1-Score, a mayor valor mejor la clasificación.

Observando los resultados de la tabla 12 podemos ver que la media se comporta de manera muy similar al sumatorio en todas pruebas. La agregación Choquet mejora ligeramente los resultados, aunque sigue obteniendo valores más bajos en los mismos dataset que el sumatorio. Estas tres agregaciones aparentemente tienen problemas con los dataset con mayor número de instancias. La agregación de Sugeno Generalizada mejora ampliamente los resultados en 3 de los 5 dataset, incluyendo uno de los dos mayores datasets. Finalmente, la agregación que sin duda obtiene los mejores resultados se trata de la mediana, es capaz de obtener buenas puntuaciones independientemente del tamaño del dataset o de su índice IR.

4.2.5.2. Versión estática

Esta versión de la función depende de los valores iniciales de sus dos variables principales, B1 y B2, los cuales no se modifican durante la ejecución del algoritmo.

La variable B1 se utiliza para acelerar el movimiento de las partículas durante las instancias iniciales del algoritmo, mientras que la variable B2 se utiliza para ralentizarlo durante las etapas finales. Ambas variables deben inicializarse desde el comienzo y por tanto es importante fijar un buen valor para ambas variables al comienzo de la ejecución.

Para empezar, hemos decidido tomar tres configuraciones iniciales distintas y comprobar sus resultados ya que, aunque seguramente no se traten de los valores óptimos, nos darán una primera impresión de los resultados que podremos obtener utilizando esta versión del algoritmo.

Para este conjunto de pruebas, utilizaremos los mismos valores prefijados para las pruebas de la versión anterior.

Dataset\Agregación	Sumatorio	Media	Mediana	Choquet	Sugeno
Ecoli 1	0.45733491	0.48634525	0.85503326	0.45324453	0.68816508
Ecoli 3	0.50463417	0.59662485	0.93408593	0.54934645	0.48150223
Glass 6	0.38025873	0.47131211	0.89307263	0.48875832	0.3296445
Wisconsin	0.12910021	0.13918942	0.7837482	0.1650487	0.7843806
Yeast 3	0.3376148	0.34606697	0.93488296	0.36299387	0.09675877

Tabla 13 F1-Score para B1 = 0.1 y B2 = 0.9

Los resultados que se pueden ver en la tabla 13 son los esperados, se puede observar una ligera mejora general, independientemente de la agregación empleada, a excepción de la agregación de Sugeno Generalizada, la cual ha empeorado la mayoría de sus puntuaciones.

Dataset\Agregación	Sumatorio	Media	Mediana	Choquet	Sugeno
Ecoli 1	0.46261806	0.52003297	0.85110816	0.50026475	0.81631772
Ecoli 3	0.50841856	0.61263632	0.93699597	0.35521751	0.90763983
Glass 6	0.38477517	0.42298619	0.87630701	0.49598433	0.30586575
Wisconsin	0.13429063	0.10547425	0.78723551	0.09505333	0.78223796
Yeast 3	0.32378663	0.41656162	0.93593925	0.38178357	0.08467506

Tabla 14 F1-Score para B1 = 0.2 y B2 = 0.8

En rangos generales, de nuevo los datos de la tabla 14 muestran una mejoría general, especialmente frente a los dataset con mayor número de instancias. De nuevo se puede observar una bajada de rendimiento en la agregación de Sugeno Generalizada, aunque no tan severa como en la tabla 2, lo cual nos puede indicar que estemos más cerca de los valores óptimos de B1 y B2.

Dataset\Agregación	Sumatorio	Media	Mediana	Choquet	Sugeno
Ecoli 1	0.40498475	0.48538683	0.83149783	0.45713287	0.69309736
Ecoli 3	0.54030423	0.546625068	0.9136877	0.48602583	0.84956239
Glass 6	0.30433287	0.40826304	0.87656154	0.46979971	0.41310169
Wisconsin	0.1088536	0.1176648	0.7845037	0.09901455	0.83138875
Yeast 3	0.26669654	0.33325541	0.92577654	0.36348439	0.11010142

Tabla 15 F1-Score para B1 = 0.3 y B2 = 0.7

En esta ocasión, en la tabla 15 se puede observar una bajada generalizada en las puntuaciones de las diferentes agregaciones, aunque con alguna que otra mejora puntual. Teniendo en cuenta los resultados de las tablas 14 y 15, podemos asumir que el valor óptimo de B1 y B2 se encuentra entre los usados en estas. Teniendo en cuenta esto, más adelante realizaremos una búsqueda más profunda de esos valores.

4.2.5.3. Versión Dinámica

En la tercera versión utilizamos la versión dinámica de la formula H. Esta versión toma como valores iniciales de B1 y B2 y los actualiza en cada iteración del algoritmo. La actualización de estos valores depende directamente del clúster de mayor tamaño en cada estado de la ejecución.

Mientras el clúster de mayor tamaño se inferior a un porcentaje del tamaño total del sistema, el valor de B1 y B2 ira disminuyendo hasta llegar a un valor mínimo cercano a 0. De este modo estamos fijando una cota inferior al producto de las masas. Esta cota inferior será igual al valor de B1, teniendo B1 > 0, mientras que en la multiplicación esta cota inferior siempre es 0. Utilizando esta cota se consigue acelerar la fusión de las partículas más pequeñas durante los estados iniciales de la ejecución.

Por el contrario, cuando el valor del clúster mayor supera el porcentaje fijado, el valor de B1 y B2 va aumentando en cada ejecución hasta alcanzar un valor máximo de 1. Utilizando el valor de B2 estamos fijando una cota superior a la multiplicación . Esta cota superior ayuda a ralentizar la fusión de los últimos clústeres.

Utilizando la versión dinámica, hemos realizado un nuevo conjunto de pruebas utilizando los valores prefijados en las pruebas anteriores.

Dataset\Agregación	Sumatorio	Media	Mediana	Choquet	Sugeno
Ecoli 1	0.41641423	0.63514889	0.85922656	0.84618761	0.86238249
Ecoli 3	0.36475079	0.77486991	0.93978759	0.9069931	0.94150316
Glass 6	0.32921327	0.49570491	0.91797725	0.84709783	0.92155489
Wisconsin	0.12323512	0.7743836	0.77590441	0.71543306	0.79093538
Yeast 3	0.34192466	0.24493543	0.94032274	0.32693981	0.34090751

Tabla 16 F1-Score para la versión dinámica

Observando los resultados de la tabla 16 se puede ver que utilizar la versión dinámica de la función H mejora los resultados los datasets de mayor tamaño, comportándose así del mismo modo que la versión estática de esa misma función.

Los resultados obtenidos utilizando tanto el sumatorio como la media, mejoran ligeramente respecto a la versión anterior en la mayor parte de los dataset. La mayor mejoría se puede apreciar cuando se utiliza tanto la función Choquet como la Sugeno Generalizada. La función Sugeno Generalizada obtiene los mejores valores en los 4 primeros dataset, pero obtiene malos resultados en el último. Por el otro lado, la mediana sigue obteniendo los mejores resultados de forma global, especialmente si comparamos las puntuaciones del ultimo dataset.

4.2.5.4. Comparación del rendimiento de las diferentes versiones

Tras realizar este primer conjunto de pruebas, podemos sacar un primer conjunto de conclusiones rápidas. La ilustración 2 muestra que el rendimiento de las tres versiones del algoritmo es bastante similar. En tres de los 5 dataset los resultados son casi idénticos, siendo estos los dataset Ecoli1, Ecoli3 y Wisconsin. La versión 1 del algoritmo se queda algo atrás a la hora de clasificar el dataset Yeast, mientras que tanto la versión 1 como la 2, son superadas por la versión 3 en el dataset Glass6.

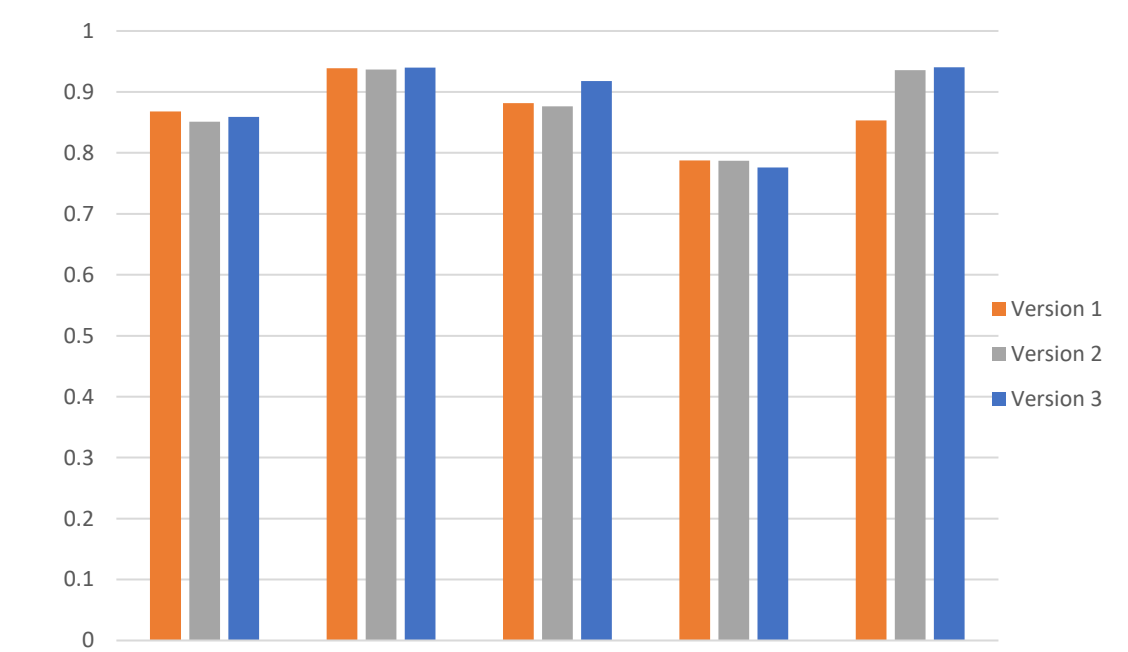


Ilustración 2 F1-Score para las primeras pruebas según la versión utilizada

Por otro lado, en esta sección hemos utilizado distintas funciones de agregación para sustituir al sumatorio. De todas las funciones empleadas, los resultados de la ilustración 3 muestran que la función que mejor rendimiento ofrece es la mediana.

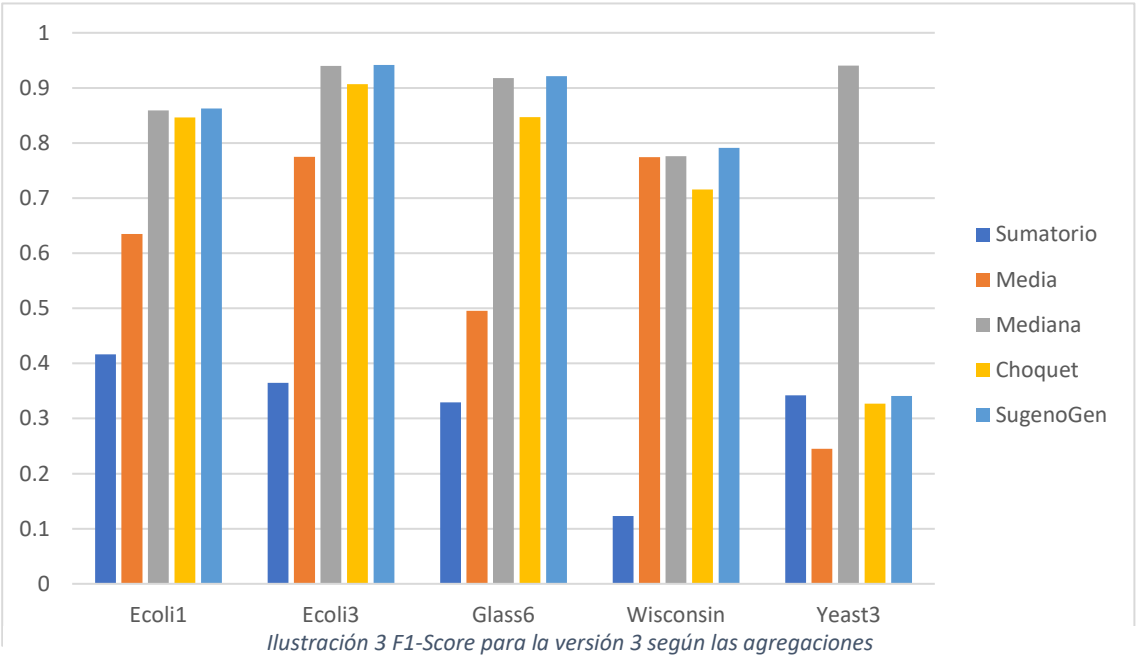


Ilustración 3 F1-Score para la versión 3 según las agregaciones

Recordemos que el dataset Yeast 3 y Ecoli 3 son los dataset con mayor índice IR, es decir, son los dataset con mayor desbalance en los datos. Teniendo esto en cuenta, queda claro que la mediana funciona mejor frente a datasets con mayor desbalance en los datos.

Del mismo modo, el dataset Yeast 3 dispone de un mayor número de instancias en comparación. A lo largo de todas las pruebas, todas las funciones de agregación han tenido problemas con los datasets de mayor tamaño, especialmente el mencionado, a excepción de la mediana y en ocasiones la Sugeno Generalizada.

Dicho esto, por el momento podemos concluir que la función de Sugeno Generalizada funciona especialmente bien con los dataset de menor tamaño, sin embargo, la mediana obtiene buenos resultados independientemente del tamaño e índice IR del dataset.

Una vez hemos seleccionada la función de agregación optima y la versión del algoritmo que ofrece un mejor rendimiento, vamos a intentar optimizar los parámetros que no son propios del algoritmo con el fin de mejorar todo lo posible los resultados obtenidos.

4.2.6. Optimización del esquema de toma de decisión

En el apartado anterior hemos realizado un primer conjunto de pruebas habiendo optimizado únicamente las variables internas del algoritmo. Sin embargo, existe otro conjunto de variables que pueden ser optimizadas y que pueden ayudar a mejorar el conjunto de resultados.

Algunas de estas variables son el número de árboles utilizados o el porcentaje de atributos usado en cada dataset.

Esta sección tiene como objetivo optimizar este conjunto de variables con el fin de mejorar las puntuaciones previamente obtenidas.

4.2.6.1. Número de árboles

El número de árboles es la variable utilizada para decidir el número de ramas que utiliza el esquema de toma de decisión. Cada una de las ramas del esquema de toma de decisión realiza el mismo proceso de clasificación, con el mismo algoritmo y los mismos valores para los parámetros, la única diferencia entre cada rama son los datos que emplea.

Cada rama del esquema de toma de decisión selecciona para su ejecución un subconjunto de los datos de train y un subconjunto de los atributos a emplear, de modo que los datos empleados sean diferentes en cada ejecución, simulando de este modo un comportamiento similar al de los árboles de decisión.

El objetivo de este comportamiento es la extracción de diferente conocimiento por parte de cada rama, de este modo es posible extraer información que, al utilizar todos los datos de forma conjunta, no seriamos capaces de obtener. Normalmente, aumentar el número de ramas en un esquema de toma de decisión tiende a mejorar el rendimiento obtenido de este, sin embargo, es posible aumentar el número de ramas y no observar ninguna mejora en los resultados, o mejoras tan pequeñas que no merezcan la pena el tiempo extra de computación.

Con esto en mente, hemos realizado un pequeño grupo de pruebas para observar si variando el número de árboles somos capaces de mejorar los resultados.

Dataset	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
NumArb = 5	0.84761114	0.9320653	0.91390273	0.74710153	0.89847136
NumArb = 10	0.85511579	0.94428602	0.9231657	0.76967238	0.93306623
NumArb = 15	0.85595049	0.94638855	0.92144896	0.78362818	0.93917508
NumArb = 20	0.85709335	0.94534143	0.9231657	0.78479754	0.94009081

Tabla 17 F1-Score para el método del voto

Dataset	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
NumArb = 5	0.84761114	0.9320653	0.91390273	0.74710153	0.89847136
NumArb = 10	0.85511579	0.94428602	0.9231657	0.76967238	0.93306623
NumArb = 15	0.85595049	0.94638855	0.92144896	0.78362818	0.93917508
NumArb = 20	0.85709335	0.94534143	0.9231657	0.78479754	0.93917508

Tabla 18 F1-Score para las agregaciones

Teniendo en cuenta los resultados obtenidos en las tablas 17 y 18, consideremos que el número de árboles optimo que utilizaremos para realizar las comparaciones frente a otros métodos deberá ser un número en el rango [10 – 15]. Teniendo en cuenta que al utilizar 15 árboles el resultado obtenido en el dataset Wisconsin ha mejorado de forma significante, tomaremos 15 como valor para las comparaciones frente a otros métodos a pesar de que no ofrezca mejora relevante para el resto de datasets.

Esto se debe a que la mejora encontrada al aumentar el número de árboles de 5 a 10 es significativa, los resultados mejoran en todos los dataset, especialmente en los dos últimos. Mientras que la diferencia entre los resultados obtenidos al utilizar 15 o 20 árboles apenas se aprecian frente coste computacional que conlleva aumentar el número de árboles.

4.2.6.2. Porcentaje de atributos

Al igual que en la sección anterior, utilizar únicamente un porcentaje de los atributos en cada rama del esquema de toma de decisión, puede ayudar a extraer información que de otra forma no obtendríamos. Es común encontrar dataset donde la clasificación de los datos se ve mayoritariamente afectada por un subconjunto de sus atributos, obviando la información aportada por los menos importantes.

Utilizando únicamente un porcentaje de los atributos, pretendemos evitar este comportamiento mencionado con el fin de extraer la información que puedan aportar todos los atributos.

Dataset	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
PorcAtr = 0.6	0.85555232	0.93805358	0.92063209	0.78702993	0.93407347
PorcAtr = 0.7	0.85325317	0.9375709	0.92232701	0.79024651	0.93806147
PorcAtr = 0.8	0.85437676	0.9375709	0.92398676	0.79095103	0.93908023
PorcAtr = 0.9	0.85325317	0.9375709	0.92063209	0.79154451	0.93895311

Tabla 19 F1-Score para el método del voto

Dataset	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
PorcAtr = 0.6	0.85555232	0.93805358	0.92063209	0.78702993	0.93407347
PorcAtr = 0.7	0.85325317	0.9375709	0.92232701	0.79024651	0.93806147
PorcAtr = 0.8	0.85437676	0.9375709	0.92398676	0.79095103	0.93908023
PorcAtr = 0.9	0.85325317	0.9375709	0.92063209	0.79154451	0.93895311

Tabla 20 F1-Score para las agregaciones

Las tablas 19 y 20 muestran un resumen de todas las pruebas anteriores para el porcentaje de atributos usado.

Podemos observar que ambas tablas son exactamente idénticas. Teniendo en cuenta los resultados obtenidos, a la hora de comparar nuestro método con otros ya establecidos, utilizaremos el 80% de los atributos para cada ejecución. Esto se debe a que los resultados muestran que este valor ofrece el mejor rendimiento en 3 de los 5 dataset, mientras que no obtiene resultados muy alejados de los mejores en los dataset restantes.

4.3. Comparaciones

En esta última subsección, presentamos los dos algoritmos actuales utilizados para la comparación de resultados y terminaremos exponiendo los resultados de dichas comparaciones.

4.3.1. LOF

Métodos que buscan outlier de forma local son de gran utilidad en dataset que contienen una gran cantidad de instancias debido al factor local que emplean. Uno de los métodos más conocidos en este grupo es el LOF.

LOF mide la desviación local de la densidad de un ejemplo dado con respecto a sus vecinos. Es local en el sentido de que la puntuación depende de cuan aislado está el objeto respecto a sus vecinos cercanos. Más precisamente, tiene en cuenta los k vecinos más cercanos, cuyas distancias son usadas para estimar la densidad local. Comparando la densidad local de una instancia con la densidad local de sus vecinos, se pueden identificar instancias que tienen una densidad local substancialmente menor que sus vecinos. Estos son considerados outliers.[21]

Aunque nosotros no usaremos dataset de grandes dimensiones, consideramos conveniente comparar el rendimiento de nuestro algoritmo frente a LOF debido a su importancia y relevancia.

4.3.2. OCSVM

Las Support Vector Machine son modelos de machine learning supervisado que utilizan diferentes algoritmos de clasificación. En esta ocasión, nos centraremos en la One-Class SVM.

Este método de clasificación tiene un funcionamiento diferente. Normalmente los métodos y algoritmos de clasificación se centran en clasificar las instancias de un dataset en diferentes clases, según las características de sus datos. Por el contrario, OCSVM únicamente se centra en encontrar una única clase bien definida y únicamente clasifica las instancias como contenidas o no dentro de esta clase.

Este comportamiento puede ser muy útil para la detección de instancias outlier. Por este motivo, compararemos el rendimiento de nuestro algoritmo frente a las OCSVM más adelante.

4.3.3. Resultados

En este apartado final del trabajo, nos disponemos a comparar los resultados obtenidos de nuestro algoritmo, frente a los métodos explicados previamente.

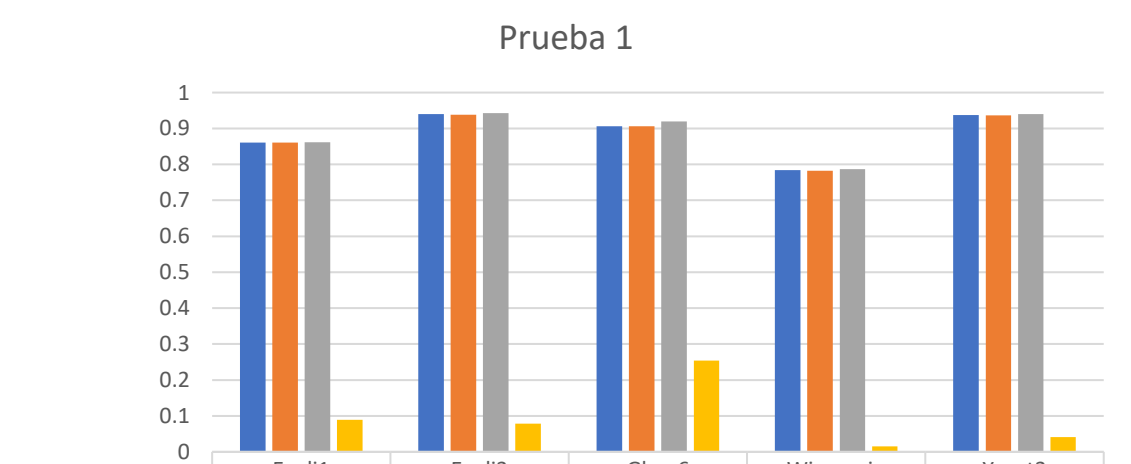
Para realizar la comparación hemos llevado a cabo 3 pruebas independientes, cada una de la hace uso de los parámetros optimizados a lo largo del trabajo, los cuales se pueden encontrar en la tabla 70.

Variable/Método	Valor/Nombre
Agregación del algoritmo	Mediana
Agregación para toma de decisión	Mínimo
Valor de la variable épsilon	0.03
Valor de la variable C	0.5
Valor de la variable K	0.12
Valor de la variable P	0.05
Valor de la variable PT	0.1
Porcentaje de train	0.4
Número de árboles	15
Porcentaje de atributos	0.8
Número de repeticiones	5

Tabla 21 Variables y métodos utilizados en la comparación

Para llevara a cabo las pruebas del método LOF y el método OCSVM, hemos utilizado la implementación presente en la librería *sklearn*. La salida de ambos métodos es un conjunto de puntuaciones, una por cada instancia del dataset, representando su predisposición a ser outliers. Teniendo esto en cuenta, se deben seleccionar como outliers las *n* partículas con mayor puntuación.

Con el fin de obtener la mejor puntuación de cada uno de los dos métodos, hemos realizado una búsqueda exhaustiva del valor de *n*, probando a clasificar desde 1 hasta el número total de instancias, siguiendo el orden de puntuaciones. De esta forma conseguimos que las puntuaciones obtenidas para ambos métodos sean lo más altas posibles. Es con estos resultados más altos con los que compararemos nuestro algoritmo.



	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
LOF	0.860888	0.939778	0.905958	0.78388	0.937398
OCSVM	0.86085965	0.93871738	0.90595883	0.78267113	0.9369181
Grav	0.86166039	0.94257009	0.91980805	0.78692841	0.9399406
Grav. Original	0.08883828	0.07834116	0.25400886	0.01558175	0.04085351

Ilustración 4 F1-Score para la primera prueba

	Ecoli1	Ecoli2	Glass6	Wisconsin	Yeast3
LOF	0.851466	0.938728	0.906748	0.793536	0.93756
OCSVM	0.85359302	0.93872624	0.90674711	0.78997307	0.93756007
Grav	0.8557179	0.94256506	0.92229504	0.79439351	0.93317645
Grav. Original	0.09374424	0.1061345	0.20943201	0.0038835	0.03221274

Ilustración 5 F1-Score para la segunda prueba

	Ecoli1	Ecoli3	Glass6	Wisconsin	Yeast3
LOF	0.858718	0.93481	0.895586	0.786164	0.938928
OCSVM	0.85282766	0.9348094	0.89558816	0.78282931	0.93844946
Grav	0.85706682	0.93762442	0.91140146	0.7841102	0.94134022
Grav. Original	0.22307461	0.10333004	0.32367995	0.01206807	0.03284799

Ilustración 6 F1-Score para la tercera prueba

Las ilustraciones 4, 5 y 6 muestran el rendimiento de nuestro algoritmo(Grav) frente a otros dos métodos ya existentes y conocidos como es el LOF y las OCSVM. También hemos incluido en la comparación los resultados obtenidos al utilizar el algoritmo gravitacional(Grav. Original) sin ninguna de las modificaciones propuestas.

Como se puede ver en las tres ilustraciones, el rendimiento del algoritmo gravitacional para la detección de outliers es bastante bajo en comparación con los algoritmos LOF y OCSVM. Sin embargo, con las modificaciones que hemos propuesto en este trabajo de fin de grado, hemos conseguido mejorar el rendimiento hasta situarlo a la par de estos dos algoritmos. De hecho, si

nos fijamos detenidamente en las tablas de resultados, podemos observar que el rendimiento general de nuestro algoritmo es ligeramente superior.

De los 5 dataset utilizados para las comparaciones, nuestro algoritmo obtiene la mejor puntuación en 12 de las 15 pruebas. Si bien es cierto que, en la mayoría de los casos, el rendimiento es mejorado por apenas unas milésimas o menos, también lo es que en el dataset Glass6 nuestro algoritmo ha obtenido un mejor rendimiento con mayor diferencia en las tres pruebas.

5. Conclusiones

En este trabajo hemos propuesto un nuevo algoritmo para la detección de instancias atípicas. La detección de datos atípicos es un problema común cuando se quiere aplicar metodologías para la extracción de conocimiento, sin embargo parte de los algoritmos utilizados para su detección requieren de un conjunto de instancias clasificado previamente, el cual puede no existir. Por ese motivo, hemos presentamos un nuevo algoritmo de detección de datos atípicos no supervisado, el cual no presenta el problema previamente mencionado.

Para la creación de este algoritmo hemos realizado diferentes modificaciones sobre el algoritmo de cluster gravitacional. Hemos estudiado el efecto al sustituir el sumatorio presente en la fórmula de atracción gravitacional, utilizando en su lugar un conjunto variado de funciones de agregación. A su vez, hemos estudiado el rendimiento al sustituir el producto de las masas de esta misma fórmula por una función H. Finalmente, hemos incluido el algoritmo en un esquema de toma de decisión para mejorar el conocimiento generado por cada una de las ejecuciones, aplicando sencillos procedimientos para la toma de decisión.

Finalmente, tras presentar el algoritmo y estudiar sus diferentes parámetros, hemos comparado su rendimiento ante la detección de instancias atípicas, usando para ello dataset con desbalanceo de clases, frente a dos algoritmos actuales, Clústering Gravitacional, Local Outlier Factor y One-Class SVM, siendo los primeros algoritmos no supervisados, mientras que el ultimo se trata de uno supervisado.

6. Referencias

[1] Brandt, S. and Brandt, S., 1998. Data analysis. Springer-Verlag.

[2] García, S., Luengo, J. and Herrera, F., 2015. Data preprocessing in data mining (Vol. 72). Cham, Switzerland: Springer International Publishing.

[3] Ben-Gal I. (2005) Outlier Detection. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA.

[4] Dixon, W. J. "Analysis of Extreme Values." The Annals of Mathematical Statistics 21, no. 4 (1950): 488-506.

[5] Barnett, V., 1978. The study of outliers: purpose and model. Journal of the Royal Statistical Society: Series C (Applied Statistics), 27(3), pp.242-250.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM computing surveys (CSUR), vol. 41 (3), pp. 1-15, jul. 2009.

[7] Sathya, R. and Abraham, A., 2013. Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence, 2(2), pp.34-38.

[8] Mandhare, H.C. and Idate, S.R., 2017, June. A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques. In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 931-935). IEEE.

[9] Cunningham, P., Cord, M. and Delany, S.J., 2008. Supervised learning. In Machine learning techniques for multimedia (pp. 21-49). Springer, Berlin, Heidelberg.

[10] Barlow, H.B., 1989. Unsupervised learning. Neural computation, 1(3), pp.295-311.

[11] Duan, L., Xu, L., Liu, Y. et al. “Cluster-based outlier detection”. Ann Oper Res 168, 151–168 (2009).

[12] Yumin Chen, Duoqian Miao, Hongyun Zhang, “Neighborhood outlier detection”. Expert Systems with Applications. Volume 37, Issue 12. 2010. Pages 8745-8749. ISSN 0957-4174.

[13] Loureiro, A., Torgo, L. and Soares, C., 2004, June. Outlier detection using clustering methods: a data cleaning application. In Proceedings of KDNet Symposium on Knowledge-based systems for the Public Sector. Bonn: Springer.

[14] W. E. Wright, Gravitational Clustering, Pattern Recognition, Pergamon Press 9, 1977, pp. 151-166

[15] Jiang Xie, Zhongyang Xiong, Qizhu Dai, Xiaoxia Wang, YufangZhang, A local-gravitation-based method for the detection of outliers and boundary points

[16] Gomez J., A new gravitational clustering algorithm

[17] Sugeno M (1974) Theory of fuzzy integrals and its applications. Tesis Doctoral, Tokyo Institute of Technology, Tokyo, Japan

[18] J. Armentia, I. Rodríguez, J. Fumanal Idocin, Humberto Bustince, M. Minárová, and A. Jurio, 'Gravitational clustering algorithm generalization by using an aggregation of masses in newton law', in New Trends in Aggregation Theory, eds., Radomír Halaˇs, Marek Gagolewski, and Radko Mesiar, pp. 172–182, Cham, (2019). Springer International Publishing.

[19] Harris, C.R. et al., 2020. Array programming with NumPy. Nature, 585, pp.357–362.

[20] KEEL-dataset citation paper: J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 (2011) 255-287.

[21] Scikit-learn citation paper: Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. : Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12: 2825-2830 (2011)

[22] Xujun Zhao, Jifu Zhang, Xiao Qin, LOMA: A local outlier mining algorithm based on attribute relevance analysis, Expert Systems with Applications 84: 272-280 (2017)

[23] Francesco Bardozzo, Borja De La Osa, Ľubomíra Horanská, Javier Fumanal-Idocin, Mattia delli Priscoli, Luigi Troiano Roberto Tagliaferri, Javier Fernandez, Humberto Bustince, Sugeno integral generalization applied to improve adaptive image binarization.

[24] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga: Outlier Detection with Autoencoder Ensembles. Proceedings of the 2017 SIAM International Conference on Data Mining (SDM). 90-98 (2017).

[25] Arthur Zimek, Ricardo J.G.B. Campello, and Jörg Sander. 2014. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. SIGKDD Explor. Newsl. 15, 1 (June 2013), 11–22.

[26] Singh, K. and Upadhyaya, S., 2012. Outlier detection: applications and techniques. International Journal of Computer Science Issues (IJCSI), 9(1), p.307.

7. Anexo

7.1. Lenguaje de programación

Un lenguaje de programación es un lenguaje formal el cual, empleando una serie de instrucciones, perime escribir un conjunto de acciones, ordenes, datos o algoritmos.

En la actualidad existen muchos lenguajes de programación, algunos comunes a diferentes usos y otros más especializados. Python, C++ o Lisp son algunos ejemplos de lenguajes de programación especializados, concretamente en el desarrollo de software para el campo de la Inteligencia Artificial.

Para la implementación de las pruebas hemos decidido emplear Python. Esto se debe a que es un lenguaje dinámico, muy potente y que cuenta con un gran conjunto de librerías que facilitaran la implementación del algoritmo.

7.2. Entorno de trabajo

7.2.1. Anaconda

Anaconda se trata de un distribuidor de los lenguajes de programación Python y R. El gran atractivo que ofrece esta plataforma es la facilidad que ofrece a la hora de la creación de entornos de trabajo e instalación de módulos y librerías.

Anaconda Navigator permite la creación de entornos de trabajo personalizados, al igual que su exportación e importación, permitiendo compartir desarrollos software con mayor facilidad. También cuenta con acceso sencillo y rápido y cómodo, no solo a todas las librerías de mayor uso en el mundo de la Inteligencia Artificial, sino también a muchos editores de texto.

7.2.2. Spyder

Spyder se trata de un entorno de desarrollo integrado en Anaconda y que esta preinstalado en Anaconda Navigator. Al diferencia de otros entornos que ofrece Anaconda, como Jupyter Notebook, Spyder permite el testeo interactivo del código, al igual que la capacidad para depurarlo.

Es debido a estas características que hemos decido utilizar Spyder en lugar de otros más user-friendly como Jupyter Notebook.

7.3. Librerías

7.3.1. Numpy

Numpy es un paquete fundamental para ciencia de la computación en Python. Es una librería de Python que provee un objeto array multidimensional, varios objetos derivados de este(como máscaras y matrices), y un conjunto de rutinas para operaciones rápidas con arrays, incluyendo matemáticas, lógicas, manipulación de dimensiones, ordenación, selección, I/O, transformaciones discretas de Fourier, algebra lineal básica, operaciones estadísticas básicas, simulaciones aleatorias y mucho más.

7.3.2. Sklearn

Scikit-learn(anteriormente scikits.learn y también conocido como sklearn) es una librería libre para machine learning para el lenguaje de programación Python. Incluye varia clasificaciones, regresiones y algoritmos de clústering incluyendo máquinas de soporte vectorial(SVM), Random forest, gradient boosting, k-means y BDSCAN, y está diseñado para ser incorporado con las librerías numéricas y científicas, Numpy y SciPy, de Python.

7.3.3. Random

Este módulo implementa un generador pseudoaleatorio de números para varias distribuciones.

7.3.4. Csv

El módulo CSV implementa clases para leer y escribir datos tabulares en formato CSV .

7.3.5. Pandas

Pandas es una herramienta de código libre rápida, potente, flexible y fácil de usar para el análisis y manipulación de datos, construida sobre el lenguaje de programación Python .

7.3.6. Matplotlib

Matplotlib es una librería exhaustiva para la creación de visualizaciones estáticas, animadas e interactivas en Python.