



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería
Informática

Aplicación de clustering de
datos educacionales en
Moodle



Presentado por Xing Long Ji
en Universidad de Burgos — 9 de julio
de 2020

Tutor: Dr. D. Raúl Marticorena Sánchez



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



Dr. D. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Xing Long Ji, con DNI X3026017E, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado aplicación de clustering de datos educacionales en Moodle.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 9 de julio de 2020

Vº. Bº. del Tutor:

Dr. D. Raúl Marticorena Sánchez

Resumen

El auge de las enseñanzas *online* ha provocado el uso de plataformas de aprendizaje virtual como Moodle. Estas plataformas almacenan información de la actividad y las calificaciones de los alumnos.

El proyecto se basa en utilizar esta información y aplicar técnicas de minería de datos para extraer conocimiento útil para los docentes. Principalmente se van a aplicar métodos de *clustering* o análisis de grupos.

Esta herramienta integrada en la aplicación UBUMonitor, permite a los profesores realizar un análisis del agrupamiento con distintas características de los alumnos aplicando algoritmos de *clustering* sobre los datos de Moodle.

Descriptores

Analítica de aprendizaje, analítica visual, minería de datos, *clustering*, Moodle, Java, JavaFX

Abstract

The rise of online teaching has led to the use of e-learning platforms such as Moodle. These platforms store information about the activity and the grades of the students.

The project is based on using this information and using data mining techniques to extract useful knowledge for teachers. Mainly we will apply methods of clustering.

This tool, integrated in the UBUMonitor application, allows teachers to perform an analysis with different characteristics of student by using clustering algorithms on Moodle data.

Keywords

Learning analytics, visual analytic, data mining, clustering, Moodle, Java, JavaFX

Índice general

Índice general	III
Índice de figuras	V
Introducción	1
Objetivos del proyecto	3
2.1. Objetivos generales	3
2.2. Objetivos técnicos	3
Conceptos teóricos	5
3.1. Minería de datos	5
3.2. Clustering particional	6
3.3. Reducción de dimensionalidad	12
3.4. Clustering jerárquico	13
Técnicas y herramientas	15
4.1. Técnicas y metodologías	15
4.2. Lenguajes y bibliotecas	16
4.3. Herramientas de desarrollo	18
4.4. Herramientas de documentación	18
Aspectos relevantes del desarrollo del proyecto	21
5.1. Bibliotecas de clustering	21
5.2. Selección de parámetros	21
5.3. Visualización del resultado del clustering	22
5.4. Análisis de silueta	23

5.5. Diagrama de dispersión 3D	23
5.6. Aleatoriedad de los resultados	24
5.7. Nombre de los clústeres	25
5.8. Transformación y filtrado de datos	25
5.9. Clustering jerárquico	26
5.10. Calidad de código	29
Trabajos relacionados	31
6.1. UBUMonitor	31
6.2. Moodle Miner	32
6.3. Herramientas para Moodle	32
Conclusiones y Líneas de trabajo futuras	37
7.1. Conclusiones	37
7.2. Líneas de trabajo futuras	38
Bibliografía	39

Índice de figuras

3.1. Distancia de Chebyshev en un tablero de ajedrez.	7
3.2. Análisis de silueta.	10
3.3. Representación del método del codo.	11
3.4. Abstracción de la reducción de la dimensionalidad	12
3.5. Ejemplo de un dendrograma.	13
5.6. <i>PropertySheet</i> para el algoritmo <i>KMeans</i>	22
5.7. <i>PropertySheet</i> para el algoritmo <i>FuzzyKMeans</i>	22
5.8. Gráfico de clústeres 3D de plotly.	24
5.9. Ejemplo del diagrama de dispersión 3D de highcharts.	24
5.10. Solución al nombre de los clústeres	25
5.11. Dendrograma de la biblioteca.	26
5.12. Dendrograma de implementación propia.	27
5.13. Dendrograma de la biblioteca de smile.	27
5.14. Opción de dividir el resultado en agrupaciones.	28
5.15. Análisis por Codacy.	29
5.16. Análisis por Code Climate.	29
6.17. UBUMonitor.	31
6.18. <i>Clustering</i> en Moodle Miner.	32
6.19. Monitores en Tiempo-Real de Intelliboard.	33
6.20. Dashboard de Zoola.	34
6.21. Dashboard de LearnerScript.	34
6.22. Smartklass	35

Introducción

La analítica de aprendizaje [5] se puede definir como medición, recopilación análisis e interpretación de los datos del alumnado. Normalmente este tipo de análisis se lleva a cabo en entornos virtuales de aprendizaje [2]. Con este método se pretende conseguir un mayor conocimiento sobre el aprendizaje y mejorar los entornos de enseñanza. La analítica se puede aplicar de forma individual o grupal para determinar un progreso y los conocimientos adquiridos.

Esta técnica es más fácilmente aplicable a un ámbito virtual, donde la medición y recopilación de datos es más sencilla, ya que, la mayoría de los entornos virtuales de aprendizaje llevan incorporadas herramientas para este fin. Una vez recogidos los datos se pueden aplicar diferentes técnicas de análisis tales como la predicción de valores, la división de los datos en grupos o establecer relaciones entre datos.

La minería de datos es un proceso cuyo objetivo es descubrir patrones en un conjunto amplio de datos. Se emplean métodos de inteligencia artificial, aprendizaje automático y estadística.

El agrupamiento o *clustering* es una técnica de minería de datos dedicada a agrupar un conjunto de datos en función de la similitud de estos. Esta técnica de agrupamiento se puede aplicar a diversos ámbitos como, la clasificación de animales o plantas, para reconocer enfermedades o para identificar los hábitos que tienen personas.

En este proyecto se va a emplear este método para agrupar a los alumnos basándose en los datos proporcionados por la plataforma Moodle, tales como las calificaciones, los registros y las actividades completadas. Se tratará de identificar posibles patrones entre la actividad en la plataforma de un alumno y sus calificaciones.

Este proyecto se integrará en la aplicación UBUMonitor [35], la cual extrae los datos del servidor Moodle y muestra las calificaciones y los registros del curso visualmente.

Objetivos del proyecto

A continuación se especifican los objetivos del proyecto:

2.1. Objetivos generales

- Adaptar los datos recogidos de Moodle para el *clustering*.
- Integrar varios algoritmos de *clustering* mediante el uso de bibliotecas de terceros.
- Permitir al usuario la selección de los parámetros de los diferentes algoritmos.
- Mostrar el resultado del *clustering* en tablas y gráficas.
- Implementar métricas de validación de las agrupaciones.
- Exportar los resultados del *clustering*.

2.2. Objetivos técnicos

- Aprender sobre analíticas de aprendizaje.
- Aprender las técnicas de *clustering*.
- Desarrollar una aplicación en Java utilizando JavaFX.
- Utilizar Git para el control de versiones.
- Realizar documentación empleando L^AT_EX.

Conceptos teóricos

La parte teórica principal de este proyecto es la minería de datos, concretamente la parte de agrupamiento o *clustering*.

3.1. Minería de datos

La minería de datos [31] es el proceso de encontrar patrones o relaciones en conjuntos grandes de datos. Este procedimiento se emplea para predecir valores y generar conocimiento sobre los datos.

El proceso de minería está formado por los siguientes pasos:

1. Selección de datos: elegir con qué datos se quiere realizar el proceso, esto incluye los parámetros del algoritmo.
2. Transformación de los datos: adaptar los datos de forma apropiada a la minería, esto incluye técnicas como la normalización.
3. Aplicación de la minería de datos: ejecutar de métodos inteligentes para extraer patrones.
4. Extracción y evaluación de los patrones: identificar los patrones útiles y realizar un análisis de las asociaciones de los datos.
5. Interpretación y evaluación de los datos: validación de los datos y los patrones asociados.

3.2. Clustering particional

El agrupamiento o *clustering* [3] es una técnica que consiste en agrupar un conjunto de datos en grupos de elementos. Cada grupo o clúster está formada por objetos similares entre sí y distintos a otros de diferente clúster.

El *clustering* es un método de aprendizaje no supervisado [23], es decir, el modelo no tiene conocimiento sobre el clúster al que pertenece cada elemento. En este tipo de aprendizaje el procedimiento agrupa por las características de cada elemento. Por lo tanto, en este tipo de aprendizaje los grupos se forman en función de los datos.

Medidas de distancia

Existen varias formas de medir la distancia entre dos puntos. El tipo de distancia empleado para realizar el *clustering* influye en el resultado obtenido. Las más comunes son la distancia de Manhattan y la distancia euclidiana.

Distancia euclidiana

La distancia euclidiana o euclídea [28] entre dos puntos es la línea recta que une ambos, se deduce por el teorema de Pitágoras.

$$\sqrt{\sum (a_i - b_i)^2}$$

Distancia de Manhattan

La distancia de Manhattan [4] entre dos puntos se define como la suma de las diferencias de cada coordenada en valor absoluto.

$$\sum |a_i - b_i|$$

Distancia de Canberra

La distancia de Canberra [22] es una métrica similar a la de Manhattan. La distinción es que la diferencia absoluta es dividida por la suma de los valores absolutos.

$$\sum \frac{|a_i - b_i|}{|a_i| + |b_i|}$$

Distancia de Chebyshev

La distancia de Chebyshev o métrica máxima [24] entre dos puntos es el valor máximo de las diferencias entre cada coordenada.

$$\max(|a_i - b_i|)$$


	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1		1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

Figura 3.1: Distancia de Chebyshev en un tablero de ajedrez.

Algoritmos

Existen varios tipos de algoritmos para el *clustering*.

k-means

K-medias o *k-means* [27] es un método de agrupamiento que tiene por objetivo dividir el conjunto de datos en k subconjuntos. Los grupos son definidos por su centroide, que es el valor medio de todos los elementos.

Cada elemento se unirá al grupo cuyo centro esté más cerca. Los centros iniciales se eligen de forma aleatoria y se van ajustando a medida que se incluyen elementos. Este método trata de reducir la varianza dentro de cada grupo.

En cada iteración se ejecutan dos pasos, en el primero se asigna cada elemento a un grupo y en el segundo se actualiza el centro de cada grupo. Finaliza cuando las asignaciones no cambian.

Los parámetros necesarios para ejecutar este algoritmo son:

- k - número de agrupaciones o clústeres
- iteraciones - número máximo de iteraciones a ejecutar
- distancia - función para calcular la distancia entre dos puntos

k-means++

K-means++ [26] es un algoritmo para la selección inicial de los centroides de cada agrupamiento en el algoritmo *k-means*. Este método elige como centroides iniciales, puntos del conjunto de datos. Elige los puntos en función de una probabilidad definida respecto a los centros elegidos. $P(x) = D(x)^2$ donde $D(x)$ es la distancia del punto al centro más cercano. Una vez seleccionado los centros se ejecuta el *k-means* estándar.

x-means

X-means [20] es un algoritmo que extiende de *k-means* e incluye una estimación eficiente del número de agrupaciones. Soluciona uno de los principales problemas de *k-means*, que el número de agrupaciones es decidido por el usuario. El algoritmo comienza con un clúster y en cada iteración ejecuta el algoritmo *k-means* y con el resultado obtenido y mediante el criterio de información bayesiano determina los clústeres a dividir. Se tiene que indicar al algoritmo el máximo número de agrupaciones.

g-means

G-means es otro algoritmo extendido de *k-means* que trata de determinar el número de agrupaciones óptimo. Este algoritmo se basa en la hipótesis de que cada agrupación sigue una distribución normal. Ejecuta el *k-means* incrementando el número de agrupaciones. Es necesario indicar el número máximo de agrupaciones.

Fuzzy clustering

El agrupamiento difuso o *fuzzy clustering* [25] es un tipo de agrupamiento basado en la lógica borrosa o difusa [30]. Esto indica a grandes rasgos que existe una incertidumbre de la pertenencia de un punto a varios clústeres.

El algoritmo de *Fuzzy K-means* es una variante del *k-means*, con la diferencia de que un punto no se asigna solo a un clúster, sino que un punto tiene un conjunto de pesos que indican el grado de pertenencia a cada clúster.

Los parámetros necesarios para ejecutar este algoritmo son lo mismos que en *k-means* incluyendo el parámetro de borrosidad o *fuzziness* que determina el nivel de borrosidad del clúster, con valores más elevados implica clústeres más borrosos.

DBSCAN

El agrupamiento espacial basado en densidad de aplicaciones con ruido o *Density-based spatial clustering of applications with noise* (DBSCAN) [15] es un algoritmo basado en la densidad. Agrupa en función de los vecinos más cercanos y eliminando los puntos aislados o zonas de poca densidad.

Los parámetros necesarios para ejecutar este algoritmo son:

- ϵ - el radio de vecindad respecto a un punto
- *minPts* - número mínimo de puntos conectados por clúster

En el primer paso del algoritmo, se determinan los puntos núcleo o *core points*, que son los que tienen al menos *minPts* puntos a una distancia ϵ . Cada punto núcleo crea una zona de densidad con sus puntos vecino. Con estas zonas de densidad el algoritmo puede unir dos zonas si están cerca, aunque esta implementación es opcional. En el resultado se muestran solo las zonas de densidad, los puntos que no pertenecen a una zona de densidad no aparecen. Este es un algoritmo determinista a diferencia de los anteriores.

DENCLUE

DENCLUE (*DENSity CLUstering*) [21] es un algoritmo basado en la densidad. Este modelo esta basado en el KDE (*kernel density estimation*). El KDE es una técnica para buscar zonas densas. Este algoritmo funciona muy bien con un conjunto de datos de muchas dimensiones.

Validación

La validación del *clustering* [10] se realizará para determinar lo bueno que es el resultado del algoritmo de *clustering*. En general, los métodos de validación se pueden clasificar en 3 grupos:

1. Validación interna: evalúa las agrupaciones del *clustering* sin información externa, solo basándose en los datos utilizados. Se puede utilizar para determinar en número de agrupaciones y el tipo de algoritmo de *clustering* apropiado.

2. Validación externa: consiste en comparar los resultados obtenidos con un resultado conocido anteriormente. Se utiliza principalmente para comprobar si un número de grupos coincide.
3. Validación relativa: evalúa el *clustering* variando los parámetros de ejecución, como el número de agrupaciones en *k-means*. Esta validación puede ser utilizada también para determinar el número de agrupaciones.

En este apartado se explicarán algunos de estos métodos.

Coefficiente de silueta

Coefficiente de silueta o *silhouette coefficient* es una métrica de validación interna que mide cuán bien están agrupadas los elementos. Este coeficiente se obtiene del análisis de silueta.

El análisis se realiza a cada elemento y consiste en calcular la distancia media entre las agrupaciones y el elemento. Los valores elevados (cerca de 1) indican que el elemento está bien agrupado, valores bajos (cerca de 0) indican que el elemento está entre dos agrupaciones y valores negativos indican que el elemento está colocado en el grupo incorrecto. El resultado del análisis se puede reflejar en un diagrama de barras.

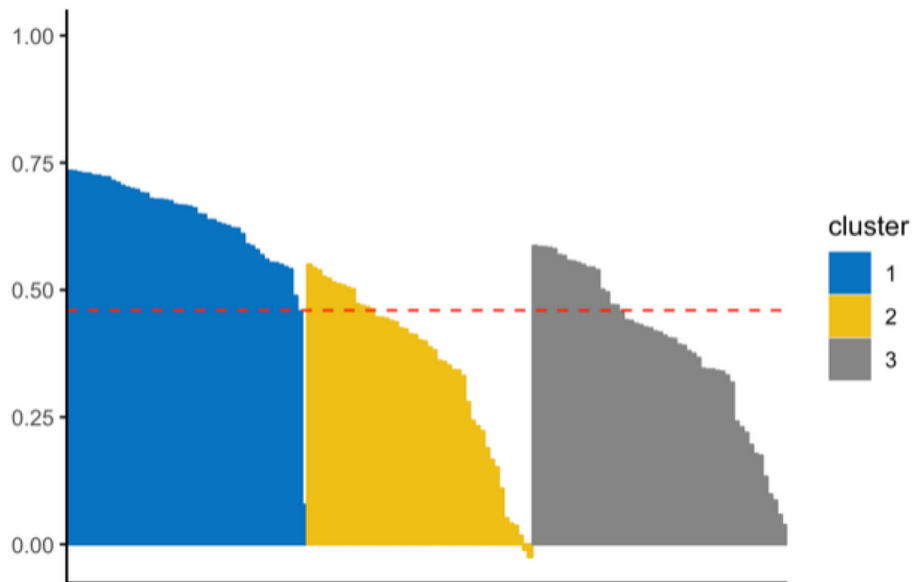


Figura 3.2: Análisis de silueta.

El coeficiente de silueta es calculado como la media de cada elemento, cuanto mayor valor es el coeficiente mejor es el *clustering*.

Método del codo

El método del codo o *elbow method* es un análisis que sirve para determinar el número de agrupaciones. Se basa en calcular la varianza dentro de cada grupo. El objetivo sería minimizar este número, cuya forma más sencilla es aumentar el número de agrupaciones. Por lo tanto a mayor número de agrupaciones menor es la varianza, cuyo mínimo sería tantos grupos como elementos. Este análisis se puede reflejar en un gráfico de líneas.

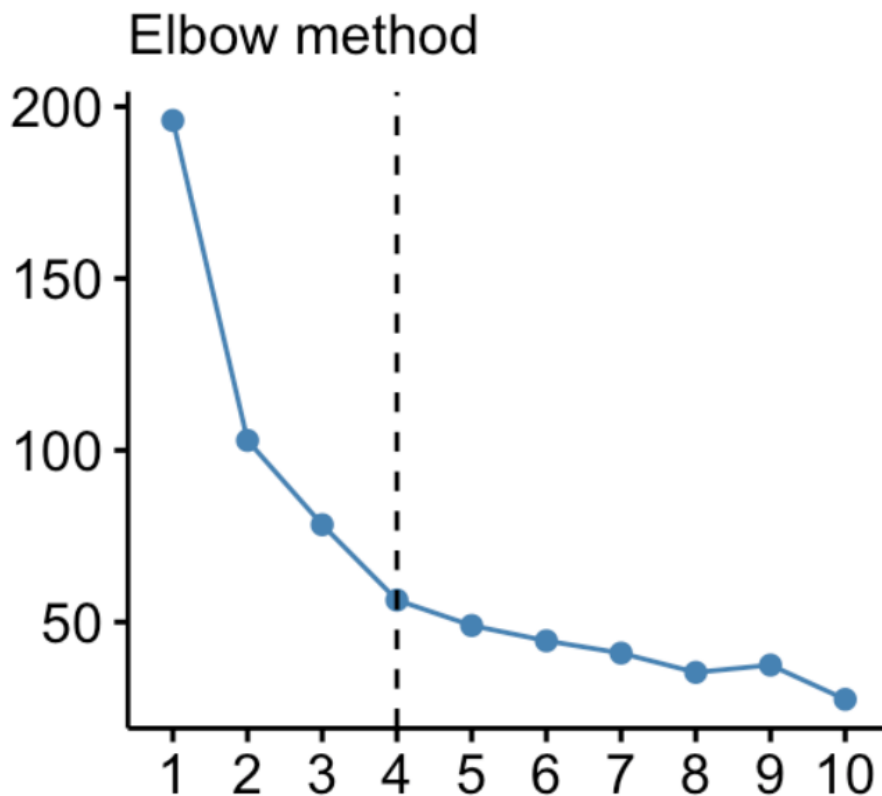


Figura 3.3: Representación del método del codo.

Observando la gráfica se puede deducir un valor óptimo de agrupaciones. Suele ser el punto donde deja de decrecer tan rápidamente, en la imagen está representada por una línea discontinua.

3.3. Reducción de dimensionalidad

La reducción de las dimensiones [34] es un proceso que trata de eliminar algunas variables no muy determinantes en el conjunto de datos. En minería de datos este método se suele aplicar antes de ejecutar algún algoritmo para reducir la cantidad de datos y obtener un mejor rendimiento.

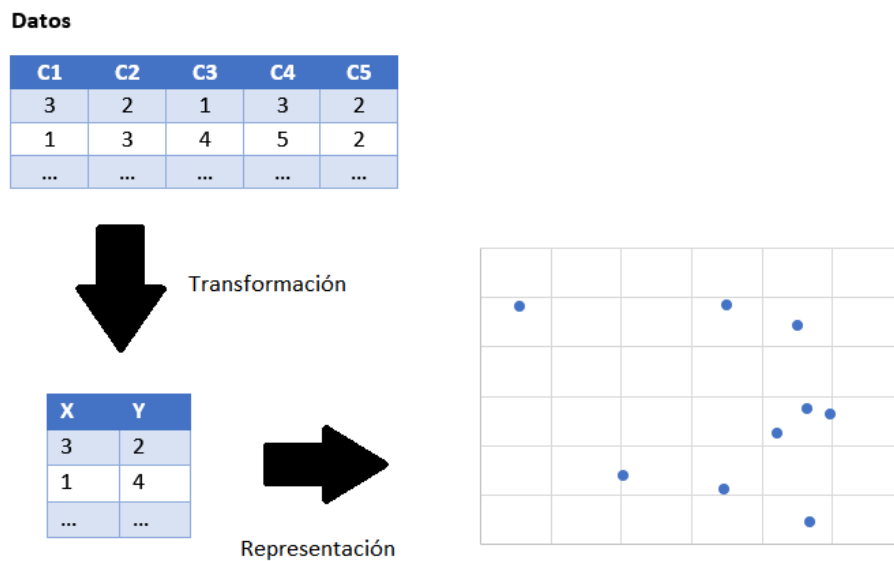


Figura 3.4: Abstracción de la reducción de la dimensionalidad

En este proyecto se ha aplicado esta técnica para generar las gráficas 2D y 3D. También se ha ofrecido al usuario la posibilidad de aplicar una reducción antes de ejecutar el *clustering*.

Existen varias técnicas, métodos y algoritmos para reducir la dimensionalidad.

Análisis de componentes principales

El análisis de componentes principales (PCA) [18] es un método estadístico que permite reducir la complejidad de espacios muestrales con muchas dimensiones. Este método está clasificado como aprendizaje no supervisado.

De forma resumida, este método calcula las componentes principales mediante una combinación lineal de las variables originales.

t-SNE

Otro algoritmo para reducir la dimensionalidad es el t-SNE [33]. Es un algoritmo de aprendizaje automático y no lineal a diferencia del PCA. Este algoritmo fue diseñado para reducir conjuntos de datos de muchas dimensiones a dos o tres dimensiones para su representación gráfica.

En este caso se asigna a cada elemento de conjunto original un punto bidimensional o tridimensional de tal manera que los elementos similares son modelados por puntos cercanos y elementos disimilares por puntos lejanos.

3.4. Clustering jerárquico

El *clustering* jerárquico [16] es otra forma de agrupar un conjunto de elementos. Existen dos tipos, en función del tipo de construcción de la jerarquía, que pueden ser *bottom-up* o *top-down*. El resultado obtenido suele representarse en un dendrograma.



Figura 3.5: Ejemplo de un dendrograma.

Los métodos *bottom-up* o aglomerativos comienzan con cada elemento en un clúster, por lo que hay tantos clúster como elementos. En cada iteración se cogen los dos clústeres mas parecidos, hasta que todos los elementos pertenezcan a un clúster. Con esto se obtiene el dendrograma. Para determinar los dos clústeres mas parecidos hay que utilizar la medida de distancia entre clústeres, que se describen posteriormente. Existen varios tipos de medida para esto.

Los métodos *top-down* o divisivos comienzan con todos los elementos en un clúster y se va dividiendo, repartiendo los elementos del clúster en dos clústeres. Esta división se puede hacer mediante un algoritmo de *clustering* normal como *k-means*. Este método es mas rápido que el aglomerativo.

Distancia entre clústeres

Existen distintas opciones para calcular la distancia entre dos clústeres al realizar el clúster jerárquico:

- Distancia entre las medias (centroides): es calculada como la distancia entre los centros de cada clúster.
- Las dos instancias más cercanas (*single linkage*): es calculada como la mínima distancia entre dos puntos de cada clúster.
- Vecino mas lejano (*full/complete linkage*): es calculada como la máxima distancia entre dos puntos de cada clúster.
- Media de las distancias entre cada par de instancias (*average linkage*): es calculada como la media de las distancias entre cada par de puntos de cada clúster.
- Criterio de *ward*: este método minimiza la varianza total dentro de cada clúster. Se calcula para cada par de clústeres.

Técnicas y herramientas

En este apartado se detallarán las técnicas y herramientas usadas durante el desarrollo del proyecto.

4.1. Técnicas y metodologías

Metodología SCRUM

Para la gestión del proyecto se ha seguido la metodología ágil SCRUM [32], aplicando una estrategia de desarrollo incremental. El flujo de trabajo está basado en *sprints*, que son los periodos en los que se realizan las tareas. Al final de cada *sprint* se presentan los avances logrados y un resultado intermedio del proyecto desarrollado. Se realizan revisiones y reuniones al final de cada *sprint*, en estas reuniones se especifican las metas para el próximo *sprint*.

Control de versiones

Se ha empleado la herramienta Git [29] para el control de versiones. Para el uso de esta se ha empleado la herramienta incluida en eclipse.

Se ha usado como repositorio de *hosting* GitHub y la extensión ZenHub para la gestión ágil.

Codacy

Codacy es una herramienta de análisis automático de código. Se emplea para detectar código duplicado y *code smells*, entre otros. Se ha utilizado esta herramienta para asegurar una calidad de código adecuada.

Code Climate

Code Climate es una herramienta similar a la anterior que evalúa la mantenibilidad del código de un programa. Para ello detecta la duplicidad y la complejidad del código. Se ha empleado esta herramienta para garantizar un buen código.

4.2. Lenguajes y bibliotecas

Java

Java [19] es un lenguaje de programación multiplataforma y orientado a objetos. Se ha empleado este lenguaje porque la aplicación donde se integra este proyecto está desarrollada en Java.

JavaFX

JavaFX [1] es un conjunto de paquetes gráficos y herramientas para el diseño y despliegue de aplicaciones, que funcionan en diversas plataformas al igual que Java. Esta tecnología es accesible desde una aplicación desarrollada en Java.

Soporta la personalización de los componentes gráficos mediante el uso de hojas de estilos en cascada (CSS) [6]. Este simple sistema se puede modificar el diseño (p. ej. los colores, la fuente y los espaciados) de los componentes.

Para el desarrollo de interfaces se emplean herramientas interactivas que permiten diseñar interfaces de usuario rápidamente.

En este proyecto se ha utilizado una biblioteca de extensión de JavaFX, denominado **ControlsFX**, la cual incluye más componentes y herramientas complementarias.

JavaScript

JavaScript (JS) [8] es un lenguaje de programación interpretado, orientado a objetos, imperativo, basado en prototipos y dinámicamente tipado. Aunque, este lenguaje es utilizado principalmente para páginas web en el lado del cliente, se puede utilizar para otros propósitos.

Se ha empleado este lenguaje junto a HTML para crear y gestionar las gráficas.

Chart.js

Chart.js es una biblioteca JavaScript de código abierto y gratuito. Se emplea para representar gráficas a partir de un conjunto de datos. Incluye una gran variedad de gráficas interactivas. Se ha decidido utilizar esta biblioteca porque el proyecto anterior ya lo usaba. Tiene licencia de uso MIT [14].

Highcharts

Highcharts es una biblioteca escrita en JavaScript basada en gráficos vectoriales escalables (SVG). Esta biblioteca proporciona una gran variedad de gráficas, tanto en 2D como en 3D. Tiene una licencia que permite el uso gratuito para un propósito no comercial. Se ha empleado esta biblioteca para el diagrama de dispersión 3D.

Apache Commons Math

Apache Commons Math es una biblioteca de Java que contiene funciones matemáticas y estadísticas no disponible en el propio lenguaje. Se ha utilizado esta biblioteca porque en el apartado de *machine learning* contiene una implementación de los principales algoritmos de *clustering*. Tiene licencia Apache [9].

Smile

Smile es una biblioteca en Java de aprendizaje automático (*machine learning*). Incluye tanto aprendizaje supervisado como no supervisado. En este proyecto se ha empleado sus implementaciones de algoritmos de *clustering*. Tiene una gran variedad de algoritmos para realizar agrupaciones. Tiene licencia LGPL [17].

Se han valorado otras bibliotecas de *clustering*, pero sus licencias han resultado ser incompatibles con el proyecto.

- Weka que tiene licencia GPL [12].
- ELKI Data Mining que tiene licencia AGPL [7].

4.3. Herramientas de desarrollo

Eclipse

Eclipse es un entorno de desarrollo integrado multiplataforma y de código abierto. Se emplea principalmente para desarrollar aplicaciones Java, aunque también soporta otros lenguajes.

Scene Builder

Scene Builder es una herramienta para la creación de diseños gráficos de JavaFX. Permite crear diferentes contenedores o *layouts*, que definen la posición y el orden de los elementos. Esta herramienta posibilita de manera sencilla la incorporación de componentes a los diversos contenedores y una visualización del resultado.

Oracle proporciona una herramienta de este estilo de forma gratuita, aunque se ha optado por usar la aplicación de Scene Builder de Gluon que permite la incorporación de bibliotecas de extensión de JavaFX.

Maven

Maven es una herramienta software para la creación y gestión de proyectos Java. Esta herramienta es especialmente útil para la gestión de bibliotecas y dependencias. Se usa para compilar, generar documentación y ejecución de pruebas.

4.4. Herramientas de documentación

L^AT_EX

L^AT_EX es sistema de elaboración de documentos. Este sistema está basado principalmente en comandos. Es usado frecuentemente para la generación de artículos y documentos científicos.

MiKTeX

MiKTeX es una distribución de L^AT_EX para Windows. Esta aplicación se encarga de gestionar los componentes y paquetes, tanto el proceso de instalación como el de actualización.

TeXstudio

TeXstudio es un editor de documentos \LaTeX . Incluye un corrector ortográfico y un resaltado de sintaxis. Esta aplicación necesita de una distribución \LaTeX para que funcione, en nuestro caso MiKTeX.

BibItNow!

BibItNow! es una extensión del navegador para citar paginas web. Incluye la opción de generar el Bibtex, que es el formato utilizado por \LaTeX para la bibliografía.

PlantUML

PlantUML es una herramienta de código abierto para crear diagramas UML. Los diagramas son generados a partir de un lenguaje simple e intuitivo. Se ha empleado esto para la realización de diagramas de clases y diagramas de secuencia entre otros, utilizando la herramienta de forma *online*.

Aspectos relevantes del desarrollo del proyecto

En este apartado se expondrán los aspectos relevantes del desarrollo del proyecto, definiendo los problemas y las decisiones tomadas.

5.1. Bibliotecas de clustering

Para el desarrollo del proyecto se necesitaban bibliotecas que implementaran algoritmos de *clustering*. La mayoría de bibliotecas en Java tenían una licencia de uso incompatible a MIT (licencia de la aplicación). También se ha investigado sobre bibliotecas de otros lenguajes como Scala y Kotlin, pero se han descartado por el desconocimiento del lenguaje. Por lo tanto, en el proyecto se ha enfocado a dos bibliotecas: la de **Apache Commons Math** en su apartado de *Machine Learning* y la de **Smile** en su apartado de *clustering*.

5.2. Selección de parámetros

Cada algoritmo de *clustering* tiene diferentes parámetros de ejecución, por lo que es necesario indicar al usuario qué parámetros. En JavaFX no se incluye ningún componente que permita introducir datos en formato clave-valor.

Se han planteado dos soluciones: la primera es añadir componentes de selección (campo de texto, cuadro de elección) de forma dinámica y en función del algoritmo seleccionado mostrar los parámetros correspondientes. Esta solución requiere programar de forma primitiva la creación y gestión de

los componentes. La otra alternativa es utilizar el componente *PropertySheet* de la librería de extensión, ControlsFX, el cual muestra los parámetros en formato clave-valor. También permite la modificación de forma dinámica de los elementos. Esto nos permite mostrar los parámetros necesarios para cada algoritmo.

KMeansPlusPlus	Número de agrupaciones	3
FuzzyKMeans	Máx. número de iteraciones	10
DBSCAN	Tipo de distancia	Manhattan
MultiKMeansPlusPlus		

Figura 5.6: *PropertySheet* para el algoritmo *KMeans*.

KMeansPlusPlus	Número de agrupaciones	3
FuzzyKMeans	Factor de borrosidad	2.0
DBSCAN	Máx. número de iteraciones	10
MultiKMeansPlusPlus	Tipo de distancia	Manhattan

Figura 5.7: *PropertySheet* para el algoritmo *FuzzyKMeans*.

Se ha optado por resolver el problema con esta última alternativa.

5.3. Visualización del resultado del clustering

Para mostrar al usuario, el resultado del *clustering* se ha implementado la visualización en forma de tabla y en una gráfica de dispersión, tanto en 2D como en 3D.

En la gráfica de dispersión, a cada punto le corresponde unas coordenadas de tamaño 2 o 3 según la dimensionalidad de la gráfica. Como los datos utilizados para realizar el *clustering* son de n dimensiones, siendo n el número de elementos seleccionados por el usuario, es necesario reducir la dimensión de los datos para que puedan ser mostrados en la gráfica.

Para resolver este problema se han empleado técnicas de reducción de dimensiones, además se ha investigado sobre las bibliotecas en Java que las implementan. Se han valorado el método de PCA y el de t-SNE y los

resultados obtenidos son bastante similares, pero la velocidad del método t-SNE es mucho menor, por lo que se ha optado por la implementación de PCA.

5.4. Análisis de silueta

La idea principal era buscar una biblioteca en Java que implementara el algoritmo del análisis de silueta. Como no se encontraron opciones válidas, se plantearon dos alternativas, buscar una biblioteca en otro lenguaje y tratar de ejecutarlo desde Java o hacer nuestra propia implementación del algoritmo.

Se optó por la segunda opción, ya que el algoritmo no es muy complejo y si utilizábamos una implementación externa habría que adaptar la estructura de los datos. Para la implementación del algoritmo se siguió el pseudocódigo de [DataNova](#).

5.5. Diagrama de dispersión 3D

Para la implementación del diagrama de dispersión 3D se han considerado diferentes opciones. Una de ellas era buscar algún componente de JavaFX que permitiera realizarlo de forma sencilla, otra opción era buscar una biblioteca en JavaScript e incorporarlo a un *WebView* de JavaFX.

La primera opción se descartó porque requería una programación de muy bajo nivel, especificando la posición exacta de cada elemento de la gráfica (ejes, cuadrículas, leyendas, puntos o esferas).

Para la segunda opción se contemplaron varias bibliotecas, una de ellas es [plotly](#) que tenía una implementación específica para clústeres y muy atractiva, se puede observar un ejemplo en la figura [5.8](#). Se descartó esta opción porque el componente *WebView* de JavaFX no soportaba el motor gráfico WebGL. Otras bibliotecas similares también tenían este problema.

Finalmente se optó por utilizar la biblioteca Highcharts que incluye el diagrama de dispersión, pero algo menos interactivo que las anteriormente valoradas. Se puede observar un ejemplo en la figura [5.9](#).

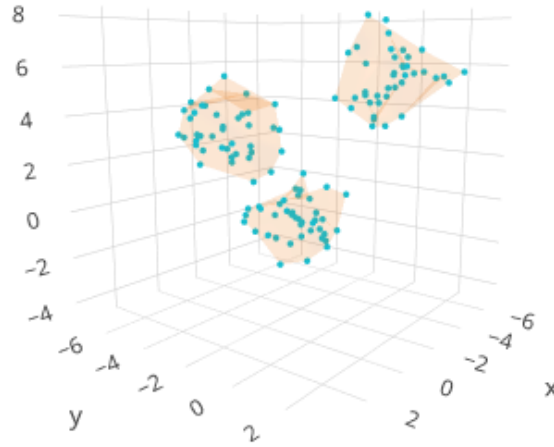


Figura 5.8: Gráfico de clústeres 3D de plotly.

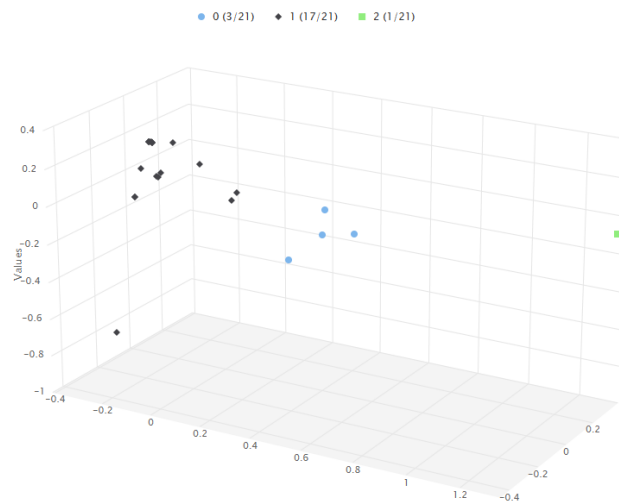


Figura 5.9: Ejemplo del diagrama de dispersión 3D de highcharts.

5.6. Aleatoriedad de los resultados

Ejecutar algunos algoritmos con los mismos parámetros y datos generan resultados distintos. Esto se debe a que algunos algoritmos utilizan un RNG

(*random number generator*) en algún paso de la ejecución. Esto producía que el resultado del *clustering* fuese mejor o peor aleatoriamente.

Para solucionar este problema se habilitó la posibilidad de que el usuario eligiera el número de iteraciones del algoritmo. Y para determinar lo bueno que es el *clustering* se empleó el coeficiente de silueta.

Al aplicar esta solución surgió un nuevo problema, la ejecución de muchas iteraciones bloqueaba el hilo principal de JavaFX y congelaba la aplicación. Para resolver esto se envió la ejecución a otro hilo utilizando la clase *Service* de JavaFX.

5.7. Nombre de los clústeres

Al ejecutar los algoritmos de *clustering* el resultado es obtenido en una lista de clústeres, donde el identificador del clúster es el índice que ocupa en la lista. En las visualizaciones del *clustering* se utiliza este número para reconocer a cada clúster.

Para resolver este problema se ha dado la posibilidad al usuario de renombrar o etiquetar el nombre de cada clúster.

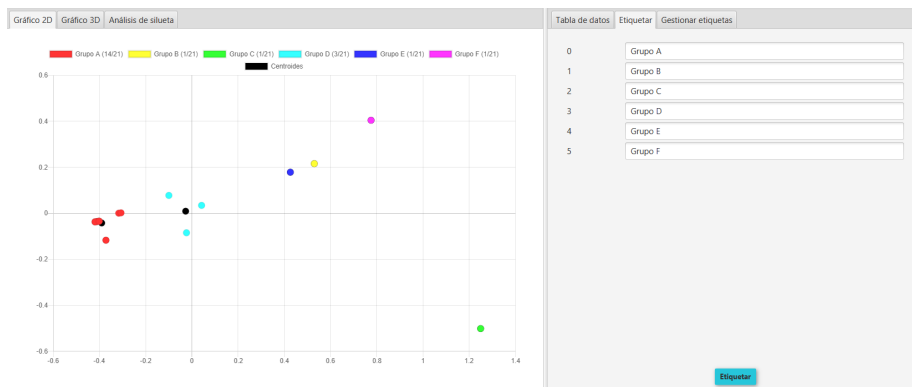


Figura 5.10: Solución al nombre de los clústeres

5.8. Transformación y filtrado de datos

Después de que el usuario haya elegido los datos con los que realizar el *clustering* hay que transformar los datos, ya que no tiene mucho sentido comparar calificaciones en un escala de 0 a 100 con el total de registros de un alumno.

Para solucionar esta cuestión, se ha realizado una normalización lineal de los datos a una escala de 0 a 1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

En cuanto al filtrado de datos, puede ser razonable eliminar aquellos valores que son constantes a todos los alumnos, aunque con algunas pruebas realizadas el *clustering* es a veces mejor con estos valores. Se ha optado por dejar esta opción a elección del usuario.

5.9. Clustering jerárquico

Para la realización del *clustering* jerárquico se planteo inicialmente utilizar la biblioteca [hierarchical-clustering-java](#), la cual daba una implementación bastante sencilla de utilizar. Una vez integrada esta biblioteca en la aplicación se procedió a valorar las posibles representaciones del resultado obtenido en el *clustering* jerárquico.

Para la visualización se contemplaron dos opciones: la primera era utilizar el propio dendrograma que ofrecía la biblioteca en Java Swing [5.11](#), la segunda opción era coger los datos numéricos del resultado y utilizar una biblioteca de gráficas para su representación.

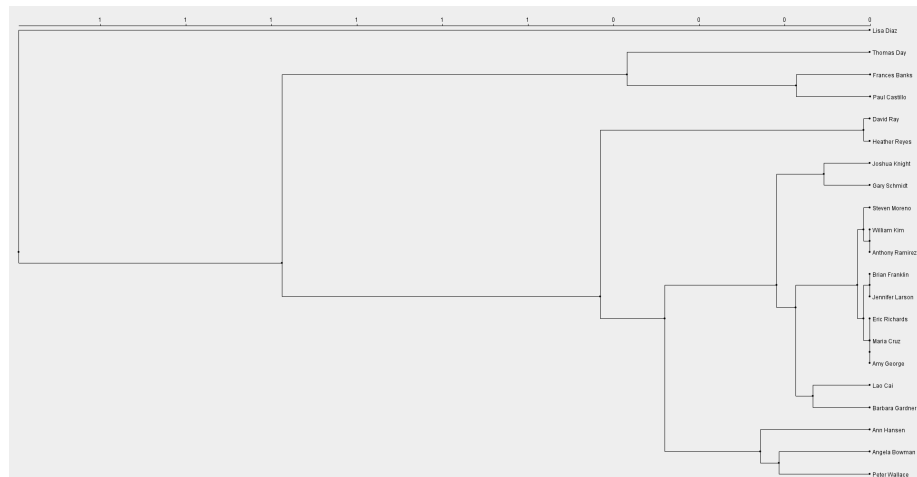


Figura 5.11: Dendrograma de la biblioteca.

Para la segunda opción se realizó una implementación empleando la biblioteca de Chart.js con la extensión [awesome](#) que ofrecía los dendrogramas.

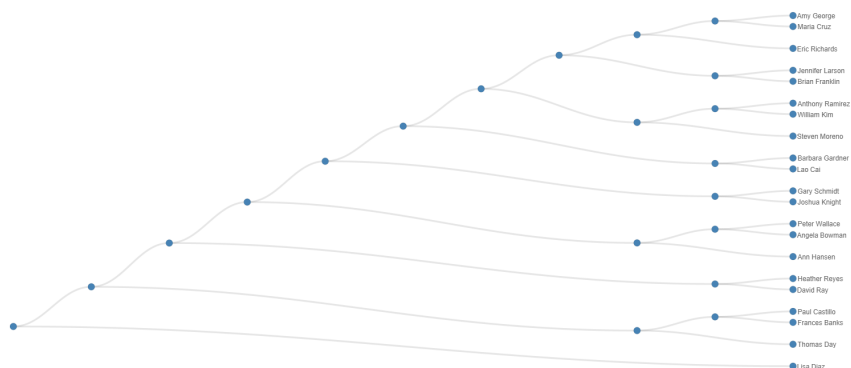


Figura 5.12: Dendrograma de implementación propia.

Inicialmente se optó por esta última opción, aunque no es muy buena, ya que no se podían ajustar los nodos en función de la distancia entre clústeres. Posteriormente se incluyó en la aplicación otra biblioteca de *clustering*, el cual incluía una implementación del *clustering* jerárquico aglomerativo.

La biblioteca **smile** permite ejecutar el *clustering* jerárquico definiendo la medida de distancia entre punto y entre agrupaciones. El resultado obtenido es mostrado en una imagen que contiene el dendrograma.

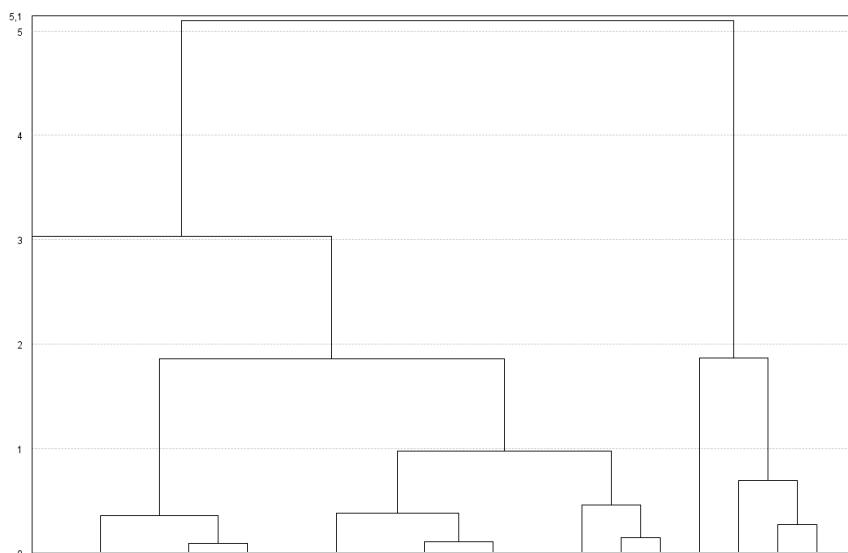
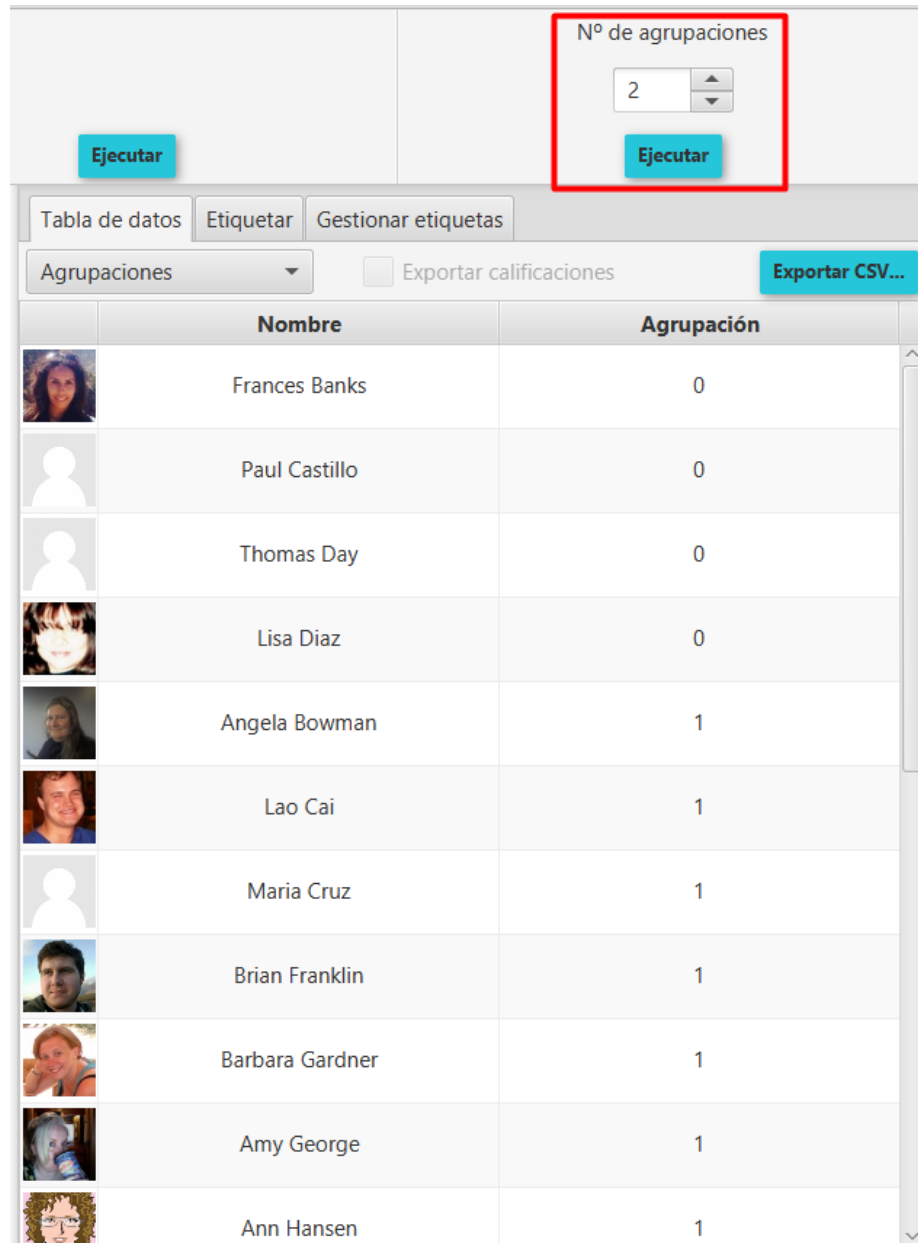


Figura 5.13: Dendrograma de la biblioteca de smile.

Se ha optado por continuar con esta opción ya que ofrece más posibilidades, aunque tiene un inconveniente, en el dendrograma no se muestra el nombre del alumno en la rama. Se ha solucionado este problema integrando la partición del *clustering* jerárquico en un número determinado de agrupaciones.














	Nombre	Agrupación
	Frances Banks	0
	Paul Castillo	0
	Thomas Day	0
	Lisa Diaz	0
	Angela Bowman	1
	Lao Cai	1
	Maria Cruz	1
	Brian Franklin	1
	Barbara Gardner	1
	Amy George	1
	Ann Hansen	1

Figura 5.14: Opción de dividir el resultado en agrupaciones.

5.10. Calidad de código

Para establecer que el código programado cumpla unos mínimos de calidad, se han empleado herramientas de análisis de código. Las herramientas utilizadas son **Codacy** y **Code Climate**. Los resultados obtenidos por Codacy.



Figura 5.15: Análisis por Codacy.

Se observa que se tienen 0% de *Issues*, el cual indica que no se tiene defectos de código o *code smells*. La complejidad de los fichero indica la complejidad ciclomática de cada clase. Y la duplicidad de código es el porcentaje de código repetido que hay.

Y los resultado de Code Climate.

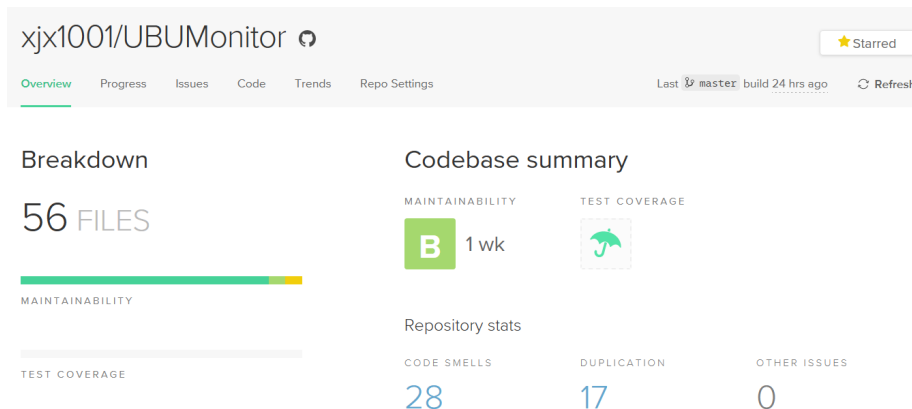


Figura 5.16: Análisis por Code Climate.

En el analisis se muestra en número de *code smells* y los bloques de código duplicado que hay. La mantenibilidad se calcula en función del tiempo necesario para corregir los defectos.

Cabe destacar que solo se han analizado los ficheros Java referentes a la parte de *clustering* y se han ignorado el resto de ficheros que no forman parte de este proyecto.

Trabajos relacionados

En este apartado se van a exponer herramientas relacionadas con la analítica de aprendizaje.

6.1. UBUMonitor

UBUMonitor [35] es el proyecto anterior donde se integra este. Es una aplicación de escritorio para la motorización de alumnos en la plataforma Moodle. Esta herramienta extrae los datos de registros, calificaciones y finalización de actividades, y los muestra en diferentes tipos de gráficas.

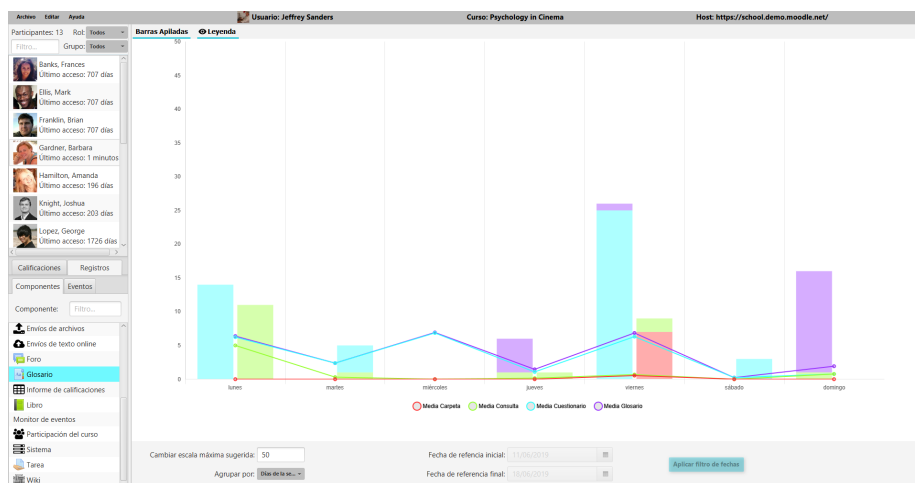


Figura 6.17: UBUMonitor.

6.2. Moodle Miner

Moodle Miner [13] es una aplicación de análisis mediante minería de datos utilizando los registros de Moodle. Utiliza una biblioteca de minería de datos, disponible en el lenguaje R, para realizar los análisis. Se incluyen técnicas de *clustering*. La herramienta permite ejecutar el algoritmo K-Means con un número determinado de agrupaciones o utilizar el número óptimo de agrupaciones calculado mediante el método del codo.



Figura 6.18: *Clustering* en Moodle Miner.

6.3. Herramientas para Moodle

Existen bastantes herramientas para Moodle que generan estadísticas en forma de tabla o gráfica [11].

Intelliboard

Intelliboard es una herramienta que funciona en cualquier LMS (*learning management system*) basado en Moodle. Ofrece monitorización en tiempo

real, puede rastrear el compromiso del usuario y la tasa de éxito del alumno. También puede ser utilizado por los administradores de Moodle para hacer un seguimiento de la actividad y el rendimiento de los profesores.

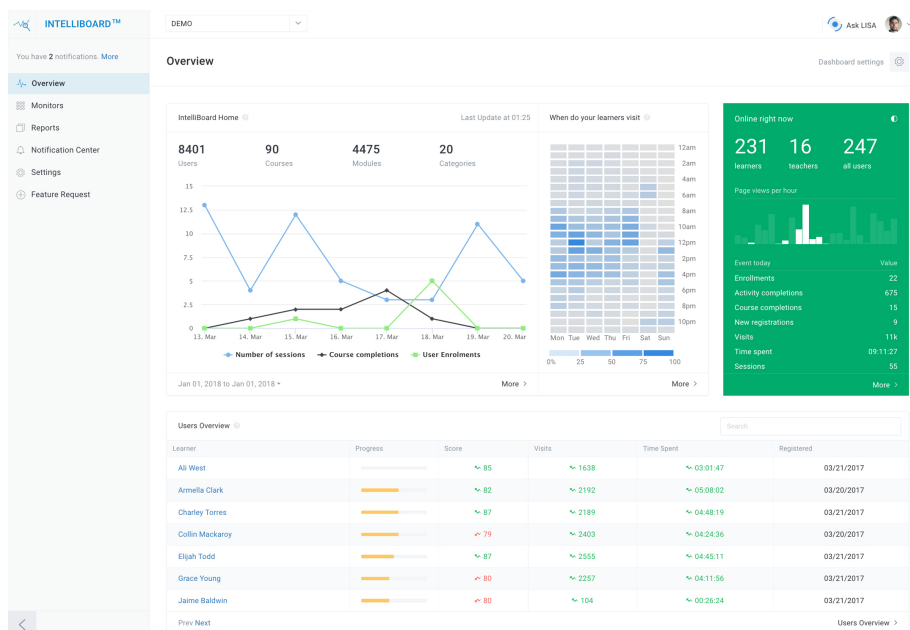


Figura 6.19: Monitores en Tiempo-Real de Intelliboard.

Zoola

Zoola es una herramienta similar a la anterior, que permite generar informes analíticos. También tiene *dashboards* personalizados.

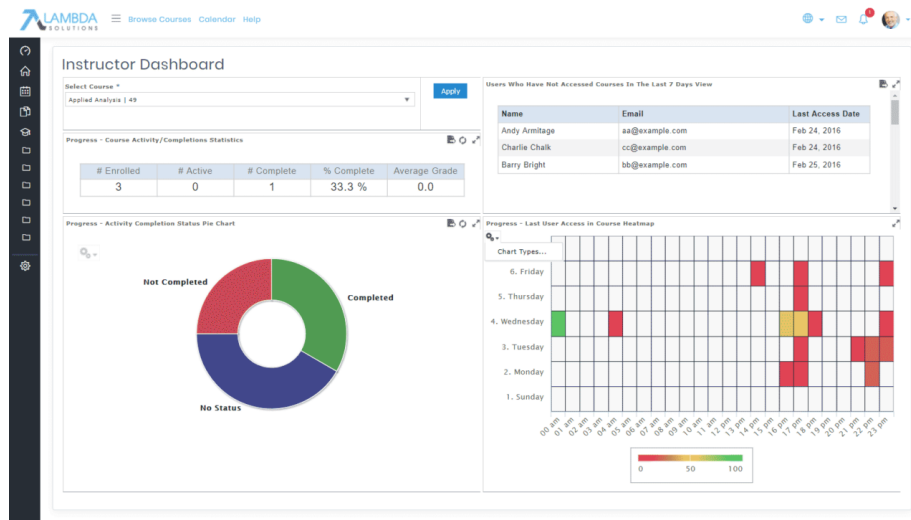


Figura 6.20: Dashboard de Zoola.

LearnerScript

LearnerScript es un *plugin* para Moodle que ofrece análisis del aprendizaje e informes. Puede ser usado por alumnos, profesores y administradores.

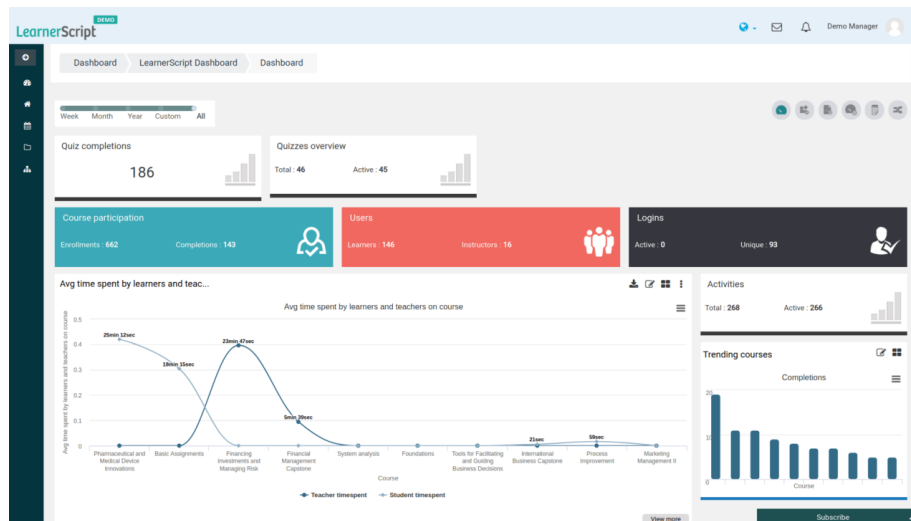


Figura 6.21: Dashboard de LearnerScript.

SmartKlass

SmartKlass es una herramienta integrable en Moodle para el análisis de aprendizaje. Utiliza algoritmos de aprendizaje automático para realizar los análisis.

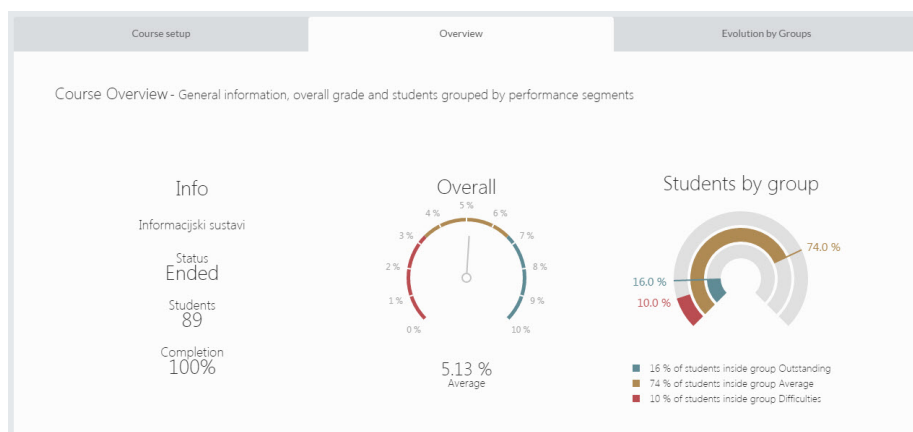


Figura 6.22: Smartklass

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Una vez terminado el proyecto, se puede concluir que los objetivos iniciales se han cumplido. De hecho, los objetivos iniciales se cumplieron bastante temprano, por lo que se incluyeron nuevos objetivos al proyecto.

Como este proyecto se integra en otro, tenía el reto de analizar y comprender cómo estaba realizado la aplicación. Además se ha realizado un desarrollo en paralelo con otro proyecto, que aun sigue en desarrollo. Esto provoca que se hayan tenido que realizar actualizaciones de la aplicación.

El aprendizaje del uso de las herramientas de desarrollo no ha sido muy complicado. La similitud de JavaFX con el diseño de interfaces en Android, algo que se ha aprendido en la carrera, ha facilitado el aprendizaje. Se ha aplicado lo aprendido de muchas de las asignaturas de la carrera, especialmente las dedicadas a la ingeniería del *software*.

Utilizando esta herramienta los docentes pueden realizar un análisis del rendimiento de los alumnos utilizando la actividad en Moodle y las calificaciones. La aplicación agrupa a los alumnos según características como: alumnos con una gran actividad y buenas calificaciones, con baja actividad y buenas calificaciones, con una gran actividad y malas calificaciones y con una baja actividad y malas calificaciones. Esto permite a los docentes tener una visión general del grupo.

Como esta herramienta está integrada en una aplicación de análisis visual, se pueden utilizar ambas funcionalidades para efectuar un análisis más profundo y sólido.

7.2. Líneas de trabajo futuras

Como posibles líneas de trabajo futuras relativas a la funcionalidad del programa son:

- Implementar la ejecución de los algoritmos en un servidor, ampliando el conjunto de algoritmos a los disponibles en otros lenguajes y plataformas.
- Añadir la opción de exportar e importar la configuración de los parámetros seleccionados para ejecutar el algoritmo de *clustering*.
- Incluir otras técnicas de analítica de aprendizaje y minería de datos como la predicción.
- Incluir mapas auto-organizados para realizar las agrupaciones.
- Realizar el *clustering* entre varias asignaturas.
- Mejorar la interfaz y la usabilidad de la aplicación.

Bibliografía

- [1] What Is JavaFX? | JavaFX 2 Tutorials and Documentation, Mar 2013. URL <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>. [Online; accessed 25. Mar. 2020].
- [2] Qué es un Entorno Virtual de Aprendizaje (EVA) -, Aug 2015. URL <https://www.aula1.com/entorno-virtual-aprendizaje-eva>. [Online; accessed 23. Mar. 2020].
- [3] ¿Que es el Clustering?, Mar 2016. URL <https://jarroba.com/que-es-el-clustering>. [Online; accessed 24. Feb. 2020].
- [4] La distancia Manhattan o la distancia euclidea, Mar 2018. URL <https://eloviparo.wordpress.com/2018/03/13/la-distancia-manhattan-o-la-distancia-euclidea>. [Online; accessed 30. May 2020].
- [5] ¿Qué son las analíticas de aprendizaje y qué utilidad tienen para la educación del siglo XXI?, Mar 2020. URL <https://ineverycrea.mx/comunidad/ineverycreamexico/recurso/que-son-las-analiticas-de-aprendizaje-y-que/d5621157-2528-9d49-5453-c639673379c3>. [Online; accessed 23. Mar. 2020].
- [6] Cascading Style Sheets, Mar 2020. URL <https://www.w3.org/Style/CSS>. [Online; accessed 25. Mar. 2020].
- [7] ELKI License – AGPLv3, Feb 2020. URL <https://elki-project.github.io/license>. [Online; accessed 22. Jun. 2020].
- [8] JavaScript, Jun 2020. URL <https://developer.mozilla.org/es/docs/Web/JavaScript>. [Online; accessed 2. Jun. 2020].

- [9] Licenses, Apr 2020. URL <http://www.apache.org/licenses>. [Online; accessed 27. Jun. 2020].
- [10] Cluster Validation Statistics: Must Know Methods - Datanovia, Jun 2020. URL <https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods>. [Online; accessed 2. Jun. 2020].
- [11] 6 Best Moodle Reporting Plugins for learning analytics in 2020 - Edwiser, Feb 2020. URL <https://edwiser.org/blog/6-best-moodle-reporting-plugins-for-learning-analytics-in-2020>. [Online; accessed 3. Jun. 2020].
- [12] Commercial applications - Weka Wiki, Jan 2020. URL https://waikato.github.io/weka-wiki/faqs/commercial_applications. [Online; accessed 22. Jun. 2020].
- [13] Gökhan Akçapınar and Alper Bayazit. Moodleminer: Data mining analysis tool for moodle learning management system. *İlköğretim Online*, 18:406–415, 01 2019. doi: 10.17051/ilkonline.2019.527645.
- [14] chartjs. License · Chart.js documentation, Jun 2020. URL <https://www.chartjs.org/docs/latest/notes/license.html>. [Online; accessed 27. Jun. 2020].
- [15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, page 226–231. AAAI Press, 1996.
- [16] Eibe Frank and Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 2016.
- [17] haifengl. smile, Jun 2020. URL <https://github.com/haifengl/smile/blob/master/LICENSE>. [Online; accessed 7. Jun. 2020].
- [18] Joaquín Amat Rodrigo J. amatrodrigo@gmail. com. Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE, Apr 2020. URL https://www.cienciadedatos.net/documentos/35_principal_component_analysis. [Online; accessed 22. Apr. 2020].
- [19] James Gosling Ken Arnold and David Holmes. *The Java Programming Language*. Addison Wesley, 1996.

- [20] Dau Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, 2000.
- [21] Hajar Rehioui, Abdellah Idrissi, Manar Abourezq, and Faouzia Zegrari. Denclue-im: A new approach for big data clustering. *Procedia Computer Science*, 83:560 – 567, 2016. ISSN 1877-0509. doi: <https://doi.org/10.1016/j.procs.2016.04.265>. URL <http://www.sciencedirect.com/science/article/pii/S1877050916302988>. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.
- [22] Jan Schulz. Canberra distance, May 2020. URL http://www.code10.info/index.php?option=com_content&view=article&id=49:article_canberra-distance&catid=38:cat_coding_algorithms_data-similarity&Itemid=57. [Online; accessed 30. May 2020].
- [23] Wikipedia. Aprendizaje no supervisado — wikipedia, la enciclopedia libre, 2019. URL https://es.wikipedia.org/w/index.php?title=Aprendizaje_no_supervisado&oldid=121459999. [Internet; descargado 24-febrero-2020].
- [24] Wikipedia. Distancia de chebyshov — wikipedia, la enciclopedia libre, 2019. URL https://es.wikipedia.org/w/index.php?title=Distancia_de_Chebyshov&oldid=121945489. [Internet; descargado 30-mayo-2020].
- [25] Wikipedia. Fuzzy clustering — wikipedia, la enciclopedia libre, 2019. URL https://es.wikipedia.org/w/index.php?title=Fuzzy_clustering&oldid=120718644. [Internet; descargado 16-abril-2020].
- [26] Wikipedia. K-means++ — wikipedia, la enciclopedia libre, 2019. URL <https://es.wikipedia.org/w/index.php?title=K-means%2B%2B&oldid=118092451>. [Internet; descargado 25-febrero-2020].
- [27] Wikipedia. K-medias — wikipedia, la enciclopedia libre, 2019. URL <https://es.wikipedia.org/w/index.php?title=K-medias&oldid=121010711>. [Internet; descargado 25-febrero-2020].
- [28] Wikipedia. Distancia euclidiana — wikipedia, la enciclopedia libre, 2020. URL <https://es.wikipedia.org/w/index.php?title=>

- [Distancia_euclidiana&oldid=126300410](#). [Internet; descargado 30-mayo-2020].
- [29] Wikipedia. Git — wikipedia, la enciclopedia libre, 2020. URL <https://es.wikipedia.org/w/index.php?title=Git&oldid=123779424>. [Internet; descargado 25-febrero-2020].
- [30] Wikipedia. Lógica difusa — wikipedia, la enciclopedia libre, 2020. URL https://es.wikipedia.org/w/index.php?title=L%C3%B3gica_difusa&oldid=123288654. [Internet; descargado 16-abril-2020].
- [31] Wikipedia. Minería de datos — wikipedia, la enciclopedia libre, 2020. URL https://es.wikipedia.org/w/index.php?title=Miner%C3%ADa_de_datos&oldid=123074000. [Internet; descargado 24-febrero-2020].
- [32] Wikipedia. Scrum (desarrollo de software) — wikipedia, la enciclopedia libre, 2020. URL [https://es.wikipedia.org/w/index.php?title=Scrum_\(desarrollo_de_software\)&oldid=123782412](https://es.wikipedia.org/w/index.php?title=Scrum_(desarrollo_de_software)&oldid=123782412). [Internet; descargado 25-febrero-2020].
- [33] Wikipedia contributors. T-distributed stochastic neighbor embedding — Wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/w/index.php?title=T-distributed_stochastic_neighbor_embedding&oldid=951308623. [Online; accessed 29-April-2020].
- [34] Wikipedia contributors. Dimensionality reduction — Wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/w/index.php?title=Dimensionality_reduction&oldid=949231135. [Online; accessed 22-April-2020].
- [35] yjx0003. UBUMonitor, Feb 2020. URL <https://github.com/yjx0003/UBUMonitor>. [Online; accessed 19. Feb. 2020].