eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Desarrollo de algoritmos para identificar alumnado con problemas de aprendizaje

Autor

Andoni Garrido Albizu

2022

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Desarrollo de algoritmos para identificar alumnado con problemas de aprendizaje

Autor

Andoni Garrido Albizu

Director/a

Francisco Vico, Ana Arruarte

Resumen

Este proyecto se enmarca en la plataforma para el aprendizaje de la programación Toolbox.Academy y tiene como objetivo la identificación temprana de problemas en el aprendizaje de su alumnado. Esta plataforma *e-learning* genera y almacena varios datos sobre la actividad de sus usuarios que abren la puerta a un análisis de que tipo de alumnos tiene y su rendimiento. El proyecto tiene varias fases diferenciadas, empezando por la investigación del estado del arte, seguido de la extracción de datos de la base de datos y el procesamiento y limpieza de los mismos. En otra fase, se estudian las variables y se seleccionan las mejores para la fase del *clustering*, donde se ha hecho un análisis de grupos. Por último, se han estudiado los grupos creados para identificar que tipo de alumnos se encuentran en cada uno y resaltar aquellos que coincidan con el objeto de este trabajo.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

En este capítulo se presenta a Digitomica como la empresa en la que se ha realizado este Trabajo de Fin de Grado y la plataforma Toolbox.Academy desarrollada por la misma. Además, se describe el objetivo del trabajo y la estructura del documento.

1.1. Digitomica y Toolbox.Academy

Digitomica es una Empresa de Base Tecnológica (EBT) creada en el Grupo de Inteligencia Computacional y Análisis de Imagen (ICAI, ETS Ingeniería Informática, Universidad de Málaga) en la que participan la Universidad de Málaga y la Asociación de Empresas Tecnológicas Vascas (GAIA). Esta organización ha sido la encargada de lanza el proyecto Toolbox.Academy [Vico et al., 2019].

Toolbox.Academy [1] es una iniciativa de naturaleza social, sin ánimo de lucro, que tiene como finalidad el desarrollo del pensamiento computacional, habilidades algorítmicas y la programación de ordenadores en las niñas, niños y jóvenes. Este objetivo se logra mediante la definición de un entorno que contextualiza los problemas con elementos sencillos, visuales e iterativos.

La plataforma se estructura en cursos, que a su vez tienen varias tareas para que el alumnado las resuelvan mediante código. Actualmente cuenta con soporte para dos lenguajes de

[1] https://toolbox.academy/es/

programación: ToyScript [Vico, 2020], un lenguaje específico de la plataforma, y Javascript. En base a estos dos lenguajes se desarrollan las tareas y sus respectivas soluciones para el proceso didáctico de los usuarios. ToyScript se centra en la enseñanza de los conceptos fundamentales de la programación, dada su sencillez. Por otro lado, JavaScript es utilizado para presentar tareas más avanzadas que requieran una mayor flexibilidad a la hora de programar.

El código introducido genera un resultado visual. Cada comando genera una secuencia de movimientos dentro de un entorno gráfico en forma de cuadrícula (mundo de Roby) donde Roby, el robot protagonista, se mueve (ver Figura 1.1).

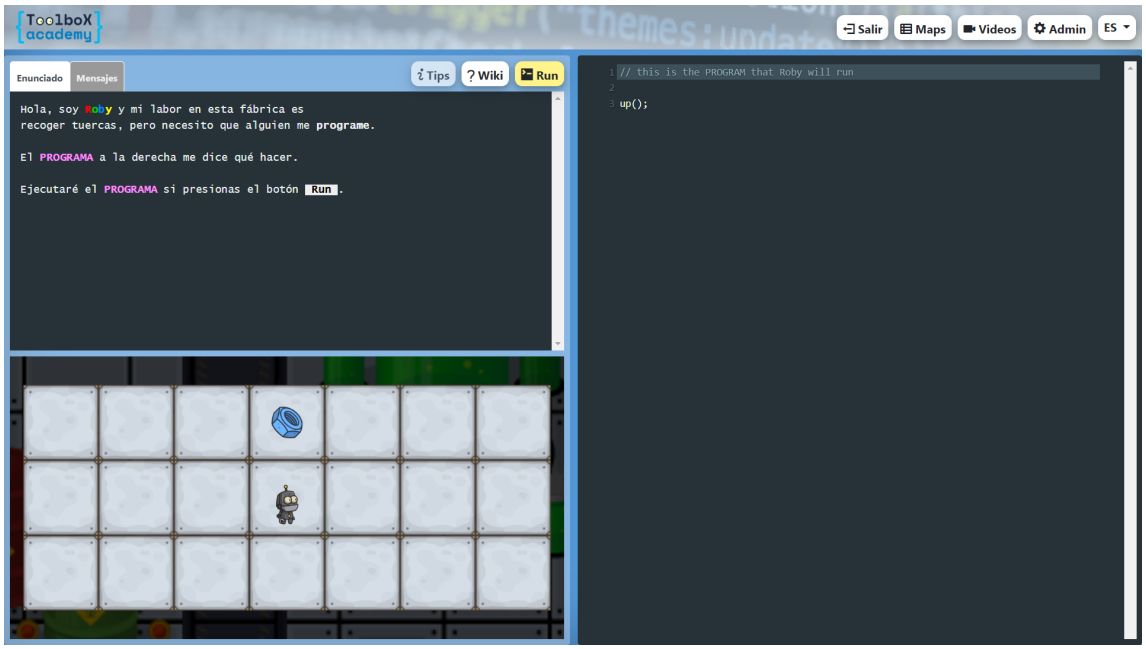

Figura 1.1: Interfaz gráfico del mundo de Roby en Toolbox.Academy.

1.2. Motivación

El principal objetivo de la plataforma Toolbox.Academy es desarrollar en los más jóvenes el pensamiento computacional y habilidades de programación. A partir de este punto, el reto de este trabajo consiste en ampliar y mejorar la experiencia de usuario mediante el análisis de la actividad de los estudiantes.

A pesar de que en la plataforma se recopilan datos de los usuarios, estos datos no son pro-

cesados ni están estructurados para posterior análisis. Es por ello que se propone habilitar esta funcionalidad.

En el amplio abanico de posibilidades que tiene la explotación de datos, en este trabajo nos vamos a centrar en la detección de problemas en el aprendizaje utilizando como criterio el rendimiento académico del alumnado en la plataforma.

1.3. Estructura del documento

Esta memoria se inicia con una introducción (1) en la que describimos brevemente la plataforma Toolbox.Academy y el reto que se aborda para la mejora de la misma.

En el Documento de Objetivos del Proyecto (2) se especifican los objetivos del proyecto y el alcance. Además se presenta la descomposición del trabajo y se describen las tareas. Junto a ello, se presenta el plan de trabajo, la metodología y el análisis de riesgos. Por último, se describe el sistema de información utilizado y las comunicaciones con los interesados.

A continuación, en la sección de Investigación (3), analizamos las diferentes herramientas y trabajos que tienen un objetivo parecido a este proyecto y especificamos las tecnologías utilizadas.

En la sección principal, Desarrollo (4), se describe el trabajo realizado en los seis paquetes de trabajo que lo componen.

En la parte final del documento está la parte de Seguimiento y Control (5) donde se presentan las desviaciones, el diagrama Gantt real, las fechas reales de los hitos y los problemas encontrados.

Por último, en la sección de Conclusiones (6), se presentan las conclusiones, las lecciones aprendidas y posibles futuras lineas de trabajo.

CAPÍTULO 2

Documento de Objetivos del Proyecto

Antes de ponerse manos a la obra, se ha hecho la planificación de las tareas a realizar en base a los objetivos establecidos. Primero, se ha analizado el alcance del proyecto junto a la estructura de paquetes de trabajo. Bajo estas directrices, se ha hecho una estimación del tiempo para cada tarea, que nos ha servido para medir la desviación de nuestro trabajo real al finalizar el proyecto (ver sección 5.1). En este marco de trabajo, también se han documentado posibles riesgos e incidencias que puedan entorpecer el proyecto. Todo esto se ha guardado en un sistema de información, descrito al final de esta sección.

2.1. Objetivos

El objeto de este trabajo es analizar que tipo de datos encontramos en la base de datos, estudiar las más útiles para un análisis del rendimiento de los estudiantes, hacer un estudio estadístico de cada variable y seleccionar el mejor algoritmo de aprendizaje no supervisado para extraer información de los datos.

Por otro lado, se define como objetivo un correcto desarrollo de la memoria en la cual se hace un control y seguimiento de los recursos empleados, se documenta el trabajo realizado y se extraen las conclusiones.

2.2. Alcance

Este proyecto gira en torno al análisis de la actividad del alumnado de la plataforma Toolbox.Academy para identificar usuarios con problemas en el aprendizaje. Para ello, estudiaremos y utilizaremos el proceso ETL (*extract, transform, load*) [SaS, 2022] con el que generaremos un *dataset* con características que nos faciliten el reconocimiento. Con estos datos, aplicaremos algoritmos de aprendizaje no supervisado y por último analizaremos los resultados de dichos algoritmos para identificar los usuarios utilizando el criterio del rendimiento académico.

Asimismo, el proyecto estará planificado con anterioridad para gestionar adecuadamente el trabajo a realizar. Con ello, se medirán los tiempos de cada fase y se registrarán los recursos utilizados. Por último, todo el trabajo desarrollado se recogerá en una memoria con una estructura adecuada, que contenga una descripción exhaustiva de los pasos que hemos seguido, los resultados obtenidos y las conclusiones. Finalmente, se expondrá y defenderá el trabajo realizado mediante una presentación.

2.2.1. Hitos

Ante un proyecto de esta magnitud, la definición de plazos es necesario para el control del trabajo realizado. Por ello, se han resaltado los siguientes hitos con el objetivo de acabar cada paquete de trabajo a tiempo (ver tabla 2.1).

Fecha	Objetivo
25-01-2022	Desarrollar planificación
25-02-2022	Base de conocimiento
23-03-2022	Dataset
01-04-2022	Implementación de los algoritmos
25-05-2022	Redacción de la memoria
24-06-2022	Presentación para la defensa

Tabla 2.1: Definición de hitos del proyecto.

2.3. Descomposición del trabajo

Consideramos de mucha utilidad descomponer el proyecto en paquetes de trabajo más pequeños mediante una estructura de descomposición del trabajo (EDT) (expresada en la figura 2.1) en la que se incluyen los paquetes de trabajo y las tareas a realizar ordenadas de forma jerárquica.

A continuación se describen los paquetes de trabajo y las tareas detalladamente:

- **Gestión**: este paquete de trabajo recoge las tareas de gestión del proyecto, como primera planificación o, según vayamos avanzado, hacer el seguimiento y control de nuestro trabajo.

- **Investigación**: antes de empezar a desarrollar, es necesario tener conocimiento del estado del arte de las tecnologías de inteligencia artificial utilizadas en educación. Es por ello que en este paquete de trabajo encontramos tareas de búsqueda de información y selección de tecnologías.

- **Desarrollo**: Una vez hayamos elegido las tecnologías que más nos ayudarán a cumplir nuestros objetivos, nos pondremos manos a la obra. En este paquete de trabajo se crea un *dataset* adecuado y se estudian, aplican y analizan algoritmos de aprendizaje no supervisado.

- **Documentación**: En paralelo a todo el trabajo, se desarrollara una memoria para documentar el proyecto. *A posteriori*, se creará la correspondiente presentación del proyecto para la defensa.

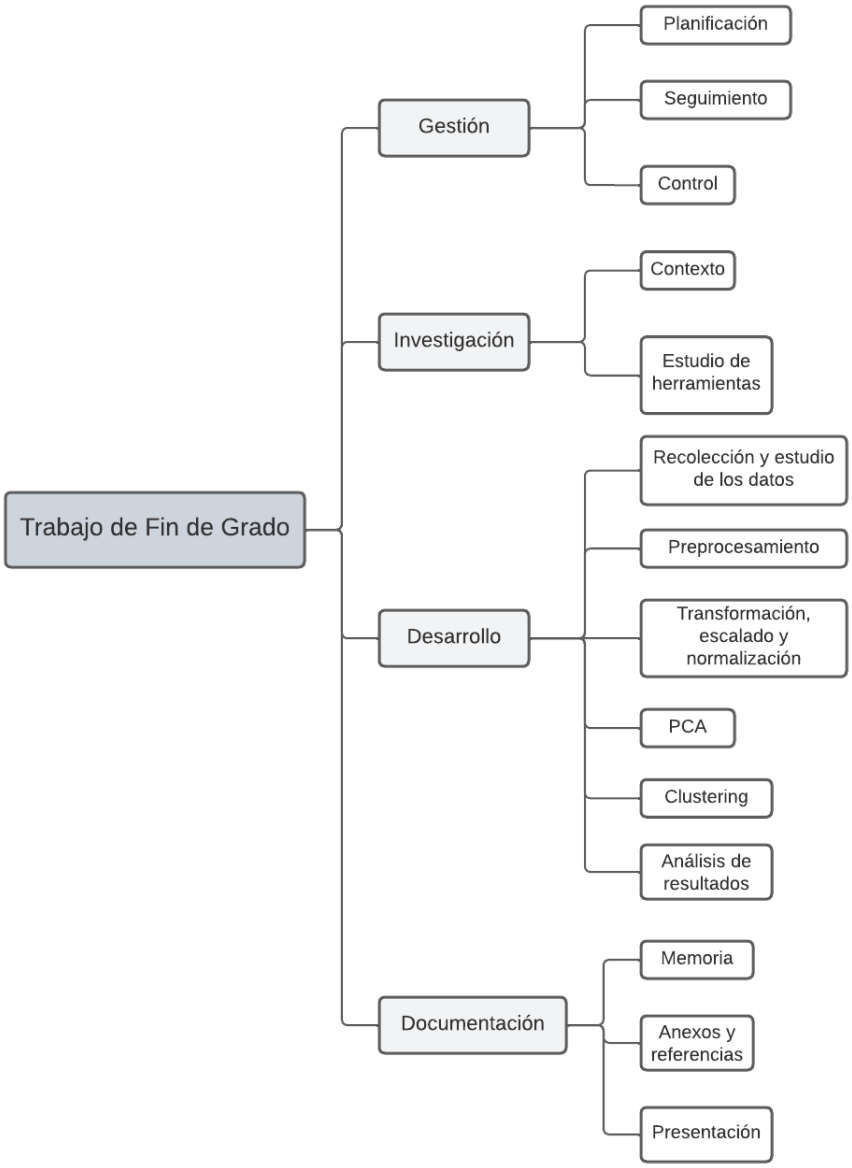

Figura 2.1: Esquema de descomposición de trabajo (EDT).

2.4. Tareas

En la tabla 2.2 se recogen las descripciones de las tareas separados por paquetes de trabajo.

Código	Tarea	Descripción
Gestión		
G.1	Planificación	Planificación del proyecto; establecer el alcance,calendario,hitos,...
G.2	Seguimiento	Control del desarrollo del proyecto, registro de las reuniones y comunicaciones, ...
G.3	Control	Evaluación de riesgos y análisis de las desviaciones.
Investigación		
I.1	Contexto	Investigar sobre tecnologías de inteligencia artificial aplicadas al entorno educativo.
I.2	Estudio de herramientas	Estudiar metodologías y herramientas disponibles para abordar el proyecto.
Desarrollo		
D.1	Recolección y estudio de los datos	Analizar la base de datos que esta a nuestro alcance y seleccionar características para crear un *dataset* acorde con nuestros objetivos.
D.2	Preprocesamiento de los datos	Analizar las variables seleccionadas para identificar valores nulos, inconsistentes y tratar variables discretas y continuas.
D.3	Transformación, escalado y normalización	Analizar la distribución de los datos y aplicar transformaciones, si fueran necesarias. A continuación, escalar y/o normalizar los datos.
D.4	*Principal Component Analysis*	Análisis de componentes principales del *dataset* para identificar las características principales, valorar una reducción de la dimensionalidad y la posibilidad de crear nuevas variables combinándolas.
D.5	*Clustering*	Estudio y aplicación de algoritmos de aprendizaje no supervisado que mejores resultados presenten. Para ello, primero se decidirá el numero de *clusters* óptimo y luego se aplicaran tres tipos de algoritmos (basado en distancias, en densidad y jerárquicos) y se seleccionara el que mejores resultado presente.
D.6	Análisis de resultados	Analizar la partición obtenida para identificar que tipo de alumnado encontramos en cada *cluster*.
Documentación		
DOC.1	Memoria	Redacción de la memoria atendiendo a los requisitos de definidos para el desarrollo de la misma.
DOC.2	Anexos y referencias	Recopilación, ordenación y dar formato a las referencias y crear anexos con información adicional.
DOC.3	Presentación	Crear la presentación para la defensa de tal forma que contenga de forma visual y resumida las gestión, las ideas más importantes y las conclusiones del proyecto.

Tabla 2.2: Descripción de las tareas ordenadas por paquetes de trabajo.

2.5. Plan de ejecución del proyecto

2.5.1. Metodología de trabajo

En este apartado especificaremos una metodología de trabajo para agilizar al máximo la forma de trabajar durante el desarrollo del proyecto y ser más productivos:

- Identificar las herramientas adecuadas para la tarea.
- Investigar y asentar una base solida de conocimiento sobre las herramientas.
- Desarrollar y documentar el código.
- Analizar los resultados y sacar conclusiones. Ante el tipo de tarea que nos encontramos, podemos anticipar que en varias ocasiones podemos estar ante el caso de volver a hacer cambios en el desarrollo para analizar diferentes resultados, es decir, una metodología de trabajo en iteraciones.
- Llevar un registro y control del tiempo que hemos empleado.
- Registrar las dudas y las respuestas para crear conocimiento.

2.5.2. Dedicación

La identificación y descomposición del trabajo va unido a la estimación del tiempo requerido para llevar a cabo cada parte del proyecto. Es por ello que en este apartado se recoge la estimación de horas inicial para cada tarea. De esta forma, finalmente es posible medir la desviación de horas en nuestro desempeño. Dicha información queda presentada en la tabla 2.3 y en la figura 2.2.

Código	Tarea	Estimación
Gestión		
G.1	Planificación	10 h
G.2	Seguimiento	15 h
G.3	Control	5 h
	Gestión:	30 h
Investigación		
G.1	Contexto	15 h
G.2	Estudio de herramientas	25 h
	Investigación:	40 h
Desarrollo		
D.1	Recolección y estudio de los datos	15 h
D.2	Preprocesamiento de los datos	30 h
D.3	Transformación, escalado y normalización	20 h
D.4	PCA	10 h
D.5	Clustering	15 h
D.6	Analisis de resultados	10 h
	Desarrollo:	100 h
Documentación		
DOC.1	Memoria	100h
DOC.2	Anexos y referencias	5 h
DOC.3	Presentación	20h
	Investigación:	125 h
	Total:	295 h

Tabla 2.3: Tabla que recoge la estimación de horas para cada tarea.

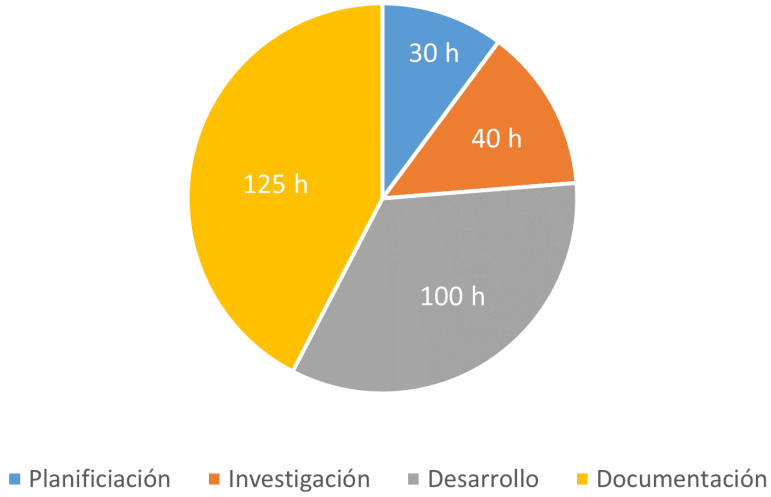

Figura 2.2: Gráfico estimación del tiempo.

2.5.3. Diagrama Gantt

Mediante un diagrama Gantt se ha visualizado las tareas programadas en una línea temporal (ver figura 2.3). De esta forma, se ha podido identificar los momentos con más carga de trabajo.

Para la generación del diagrama se ha utilizado la herramienta gratuita Gantt Proyect [1].

[1] https://www.ganttproject.biz

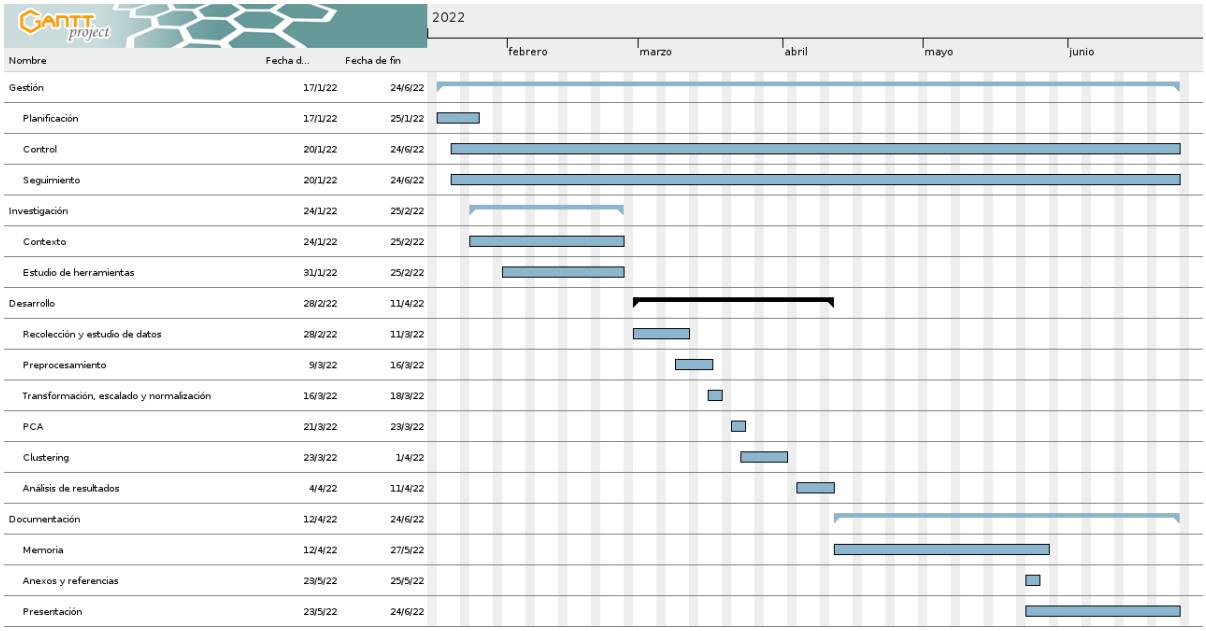

Figura 2.3: Diagrama Gantt de las tareas programadas.

2.5.4. Carga de trabajo

Analizando el diagrama Gantt (ver figura 2.3) no identificamos, *a priori*, ningún momento de acumulación de tareas, aunque entre todas las tareas el estudio de las variables (D.1), el *clustering* (D.5) o la redacción de la memoria (DOC.1) tienen más carga de trabajo.

2.6. Análisis de riesgos

Poder identificar *a priori* los posibles riesgos que pueden aparecer durante el desarrollo ayuda a tomar precauciones y, en caso de que ocurran, ser capaces de actuar más rápida y efectivamente.

No interpretar correctamente los resultados

- *Descripción:* Durante el estudio de conceptos teóricos y comprensión de las herramientas necesarias para abordar el proyecto puede que no asimilemos bien la información. Esto podría acarrean tener una base errónea y cuando haya que interpretar los resultados no hacerlo correctamente.
- *Probabilidad:* Baja

- *Impacto:* Alto
- *Posible solución:* Interiorizar y entender con detenimientos todos los conceptos teóricos y matemáticos detrás de los algoritmos y de las funciones de las librerías utilizadas.

Llegar al máximo de horas de computación en la nube

- *Descripción:* El código del proyecto ha sido desarrollado y ejecutado en Google Colab, eso implica que en alguna jornada diaria podrían restringir el acceso si sobrepasamos las cuatro horas de computo en sus maquinas. La probabilidad de este caso es baja ya que para el volumen de datos y los modelos que van a ser usados no necesitan tantos recursos.
- *Probabilidad:* Baja
- *Impacto:* Medio
- *Posible solución:* Encontrar una plataforma/computadora alternativa para ejecutar el código y/o aprovechar para realizar otras tareas que no requieran cálculos computacionales.

Perdida de información

- *Descripción:* Al tratarse de Google Colab (descripción en el apartado 3.3.1) y computación en la nube, el código se encuentra almacenado en Google Drive, es decir, online. Con lo cual, usando este servicio se asume el riesgo de que, en el caso de que los servidores de Google no estén operativos o se hayan caído, no podamos acceder al trabajo.
- *Probabilidad:* Baja
- *Impacto:* Alto
- *Posible solución:* Crear copias de reguridad periódicas y almacenarlas tanto en local como en la nube, por ejemplo, en Google Drive.

Retrasos en la planificación

- *Descripción:* requerir más tiempo para alcanzar los objetivos propuestos es un riesgo que tenemos que tener muy en cuenta, ya que existen varias razones por la cuales la planificación puede ser alterada: además de los casos mencionados con anterioridad, enfermedad, problemas personales, cuestiones académicas, entre otras.
- *Probabilidad:* Media
- *Impacto:* Medio
- *Posible solución:* Planificar el trabajo y los objetivos semanalmente y llevar un control del tiempo y la productividad.

2.7. Sistemas de información y copias de seguridad

La plataforma en la que ha sido almacenado el proyecto ha sido Google Drive [2]. Esta plataforma tiene varias características que agilizan el trabajo, entre otras resaltan la capacidad de compartir directorios, documento y Google Colab para el código. Además, la información se guarda en la nube que esta accesible a través de una aplicación web[3] o directamente desde el explorador de archivos de Linux[4], si lo configuramos correctamente.

La estructura del directorio principal ha permitido que el director de la empresa Digitomica tenga acceso al código en todo momento y que el alumno tenga documentos no compartidos como apuntes, artículos u otros recursos.

Por último, además de guardar la información en la nube, se han creado copias de seguridad periódicas y almacenadas en el ordenador personal del alumno de forma local para evitar perdida de información.

2.8. Comunicaciones

Las comunicaciones con los interesados han sido por un lado, presencialmente con la directora, Ana Arruarte, con el objetivo hacer la presentación del proyecto y definir de una linea de trabajo acorde con las exigencias de un Trabajo de Fin de Grado de la especialidad de Computación.

[2] https://drive.google.com
[3] https://drive.google.com
[4] La distribución Ubuntu 20.04 ha sido utilizada

Por otro lado, con el director de proyecto perteneciente a Digitomica, Francisco Vico, para la resolución de dudas y seguimiento del trabajo. En el caso de que la duda sea pequeña la comunicación se hará por correo electrónico. En caso contrario, se cerrará una cita para hacer reunión *online*.

2.9. Interesados (*stakeholders*)

El principal interesado en el correcto desenlace de este proyecto es el propio alumno, siendo el encargado de llevar a cabo las tareas, el control de las mismas y cumplir los objetivos propuestos.

En segundo lugar, tanto Francisco Vico como la empresa Digitomica, para que estos avances y documentación sobre la información analizada de la actividad de la plataforma Toolbox.Academy sea útil en el futuro. Seguimos con Ana Arruarte, puesta en la verificación de la correcta aplicación de los conocimientos que el alumno, Andoni Garrido, ha adquirido durante el grado para el desarrollo de este proyecto.

Por último, pero no por ello menos importante, al tratarse de un Trabajo de Fin de Grado de la especialidad de Computación, los miembros del tribunal del departamento de Ciencias de la Computación e Inteligencia Artificial.

2.10. Entregables

Con la entrega de este proyecto, como representación del trabajo que se ha realizado, se entregaran los siguientes activos:

- **Memoria** que recoge el trabajo realizado cumpliendo los requisitos previamente definidos.
- **Presentación** para la realizar adecuadamente la defensa.
- **Código** que haya sido utilizado durante el desarrollo en formato *notebook* en Google Colab.

CAPÍTULO 3

Investigación

Atendiendo a los objetivos del proyecto, se ha realizado una investigación para crear una base de conocimiento sobre el trabajo y las herramientas que se utilizan para abordar este tipo de tareas.

3.1. Contexto

La digitalización de la educación ha llevado a la creación de plataformas *e-learning* donde la docencia y los materiales académicos están en formato digital. Este avance facilita, entre otras cosas, la recopilación de datos estructurados sobre los usuarios durante su proceso lectivo. Además, el uso de estos datos en algoritmos y técnicas de aprendizaje automático han permitido utilizar métodos inteligentes que han mejorado la docencia. Entre estos métodos destacan los siguientes.

Por un lado, se encuentra el denominado *Intelligent Tutoring System (ITS)* o sistema de tutoría inteligente [Phobun and Vicheanpanya, 2010] que determina una ruta, paso a paso, a través de materiales didácticos y actividades en función del usuario. A medida que el usuario avanza, el sistema automáticamente ajusta el nivel de dificultad y provee pistas y tutoría de una forma automatizada. Una de las técnicas utilizada para ello crea un modelo probabilístico de cada estudiante en función de sus datos de rendimiento [Dašić et al., 2016]. Este proyecto no se centra en el desarrollo de esta tecnología. Sin embargo, podría ser una futura línea de trabajo.

Por otro lado, el denominado *Educational Data Minning* o conocido por sus siglas EDM [Romero and Ventura, 2007] consiste en aplicar técnicas de minería de datos y *big data* a datos de carácter educativo y su principal objetivo es entender como aprenden el alumnado y mejorar su proceso lectivo. La minería de datos es un campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos. El *EDM* además de utilizar métodos de la inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos, incorpora métodos para el análisis del aprendizaje (ver figura 3.1).

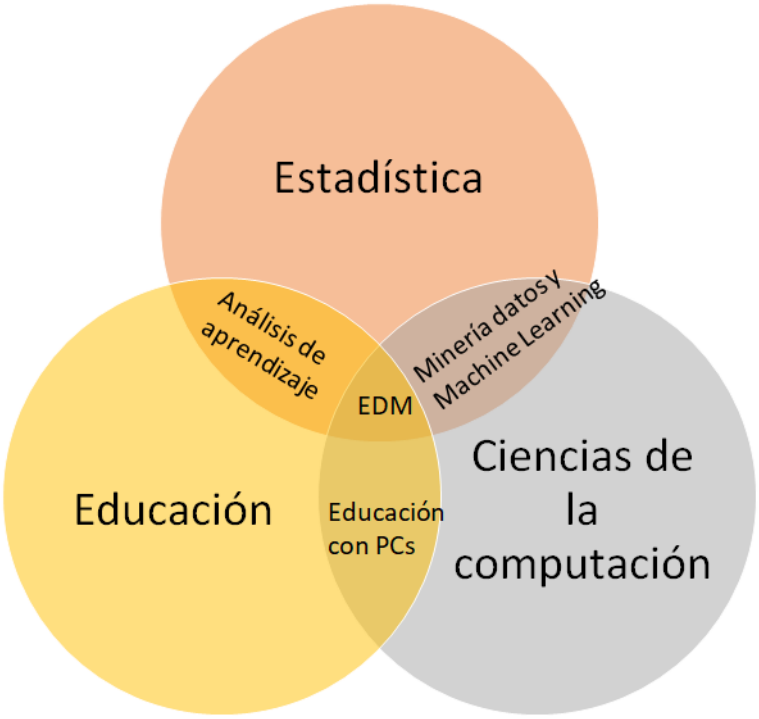

Figura 3.1: Esquema descriptivo [Moreno Sandoval, 2018] del ámbito de la minería de datos educativos [Moreno Sandoval, 2018]

En este gráfico están representados las tres áreas que forma el EDM: estadística, educción y ciencias de la computación. La conjunción de las mismas da paso a subareas reconocidas como:

- **Análisis del aprendizaje**: consiste en la medición, registro, análisis y presentación de informes sobre los datos del alumnado durante su proceso lectivo.

- **Minería de datos y Machine Learning**: conjunto de herramientas utilizadas para tratar grande volúmenes de datos y aplicación de algoritmos de aprendizaje.

- **Educación con ordenadores**: introducción de dispositivos en el proceso lectivo que habiliten el uso plataformas *e-learning*.

3.2. Metodología

Una vez situado el marco teórico, se han buscando los métodos para abordar los datos. Dentro del procesamiento de datos, la metodología más adecuada es la denominada Descubrimiento de Conocimiento en Bases de Datos, más conocida por sus siglas en ingles KDD (*Knowledge Discovery in Databases*), representada en la figura. 3.2.

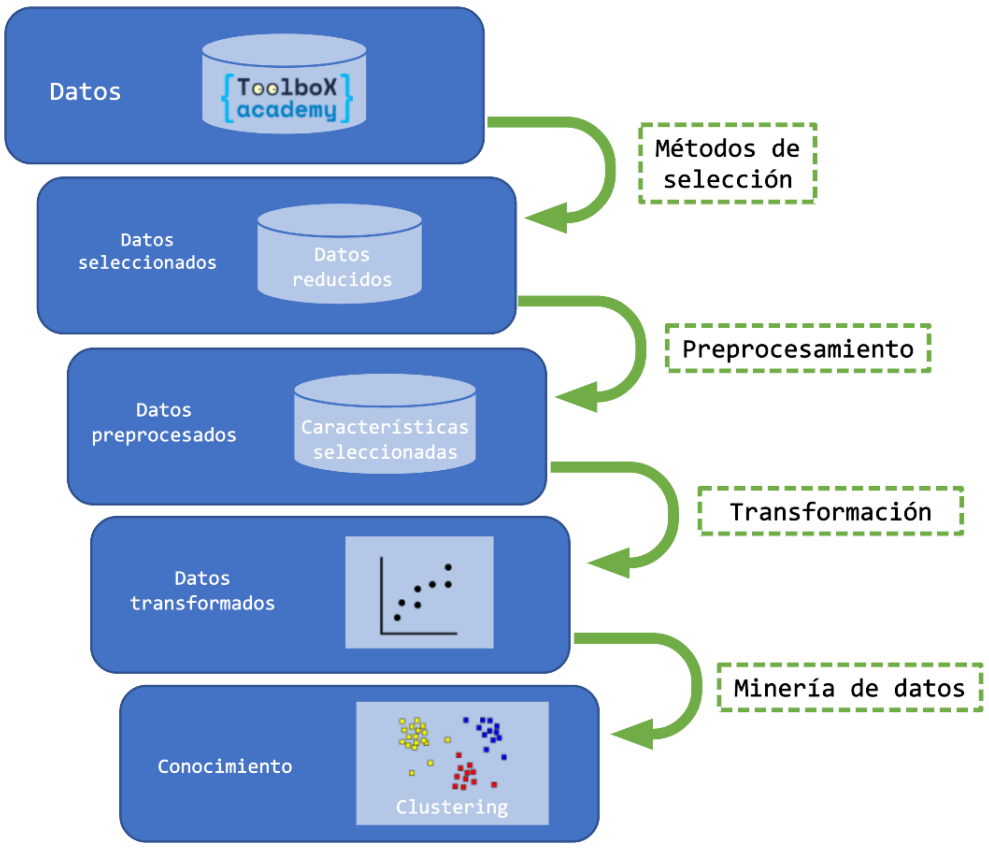

Figura 3.2: Esquema de los pasos de la metodología KDD [Regueras et al., 2019].

Esta metodología se resume en tres principales fases:

1. **Recopilación y selección de los datos**
   Esta primera fase consiste en recopilar y seleccionar los datos de la base de datos. Para ello, teniendo en cuenta la estructura de la base de datos y se identifican las características más representativas para describir al alumnado.

2. **Preprocesamiento y transformación**
   En esta segunda fase se trataran los datos con el objetivo de obtener un *dataset* apto para hacer uso de el en un algoritmo de aprendizaje. Para ello, se analiza los datos para encontrar datos inconsistentes y hacer un análisis de componentes principales o PCA por sus siglas en ingles (*Principal Component Analysis*).

3. **Minería de datos**
   En esta última fase consiste en aplicar técnicas de minería de datos para adquirir conocimiento de los datos. Los datos de los que se disponen no esta etiquetados, con lo cual, se han utilizado algoritmos de aprendizaje no supervisado, en especial algoritmos de *clustering*.

Las plataformas *e-learning* habilitan la recopilación de datos de los estudiantes mediante los *logs* o registro guardados de su actividad. Estos datos, combinados con la información sobre el alumnado y los materiales didácticos, permiten crear un *dataset* completo que habilita el aprendizaje.

3.3. Tecnologías y herramientas

Atendiendo al sentido sentido del proyecto y los objetivos definidos (2.1), se han utilizado tecnologías que habilitan métodos estadísticos y de minería de datos forma sencilla y computacionalmente eficiente. Estas han sido las tecnologías y herramientas utilizadas:

3.3.1. Google Colab

Google Colab es un producto de Google Reserch que permite programar y ejecutar Python en el navegador y tener acceso a GPUs[1]. Google Colab esta basado en el proyecto *open*

[1] https://research.google.com/colaboratory/faq.html

source Jupyter[2].

Son varias las características de este producto que han agilizado el proyecto.

- Todas las ventajas de *Jupyter notebook*. Este formato permite combinar texto con código, pudiendo incluir LaTeX. De esta forma primero, se puede generar una mejor documentación sobre el código y segundo, se ejecuta el código en el mismo *notebook*.

- La conexión con *Google Drive* permite cargar y exportar los datos directamente a nuestro sistema de información (2.7) en la nube. Además, al tratarse de almacenamiento en la nube, permite editar el *notebook* desde distintos dispositivos utilizando nuestro correo electrónico de Google, teniendo siempre una versión actualizada del trabajo.

- La visualización de los gráficos generados es una gran ventaja, ya que se pueden generar diferentes tipos de representaciones de los datos y analizarlos de forma visual sin salir del navegador.

- La facilidad para compartir el *notebook*. En el proyecto contábamos con el correo electrónico de Google del director de la empresa y de esta forma hemos podido compartir el trabajo fácilmente.

Cabe añadir que para el uso de esta herramienta es necesaria una cuenta de Gmail.

3.3.2. Python

Encontramos dos lenguajes de programación con las características que nos interesan para este proyecto, R[3] y Python[4]. Ambos cuentan con librerías para el procesamiento de datos y algoritmos ya diseñados para la minería de datos. El factor diferenciador que ha hecho que el lenguaje elegido sea Python ha sido la compatibilidad con Google Colab.

Dentro del entorno Python se han utilizado varias librerías con diferentes objetivos.

[2] https://jupyter.org/
[3] https://www.r-project.org/
[4] https://www.python.org/

Pandas

Pandas [5] es una librería de Python especializada en el manejo y análisis de estructuras de datos. Entre todas las funcionalidades, para este proyecto se han utilizado las siguientes características:

- Estructuras de datos: principalmente se ha utilizado la estructura de datos *DataFrame*. Esta clase es una estructura para datos de dos dimensiones (tabla), compuesta por *arrays* de *Numpy* y cuenta con varios métodos muy útiles para el procesamiento de los datos. Permite acceder a los datos mediante índices o nombres configurables de filas y columnas, ofrece métodos para reordenar, dividir y combinar conjuntos de datos y todo esto de forma eficiente.
- Lectura/Escritura de archivos CSV: cuenta con métodos para leer archivos CSV y guardarlos como *DataFrame* y viceversa de forma eficiente y cómoda.

Numpy

Numpy [6] es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

Sciki-learn

scikit-learn [7] es una librería de Python con herramientas para el aprendizaje automático. Para este proyecto hemos utilizado varios paquetes dentro de esta. Todos los algoritmos que utilizamos están en el paquete `cluster`, los métodos de procesamiento de datos están en el paquete `preprocessing` y las métricas e índices estan en `metrics`.

Matplotlib

Matplotlib [8] es una biblioteca para la generación de gráficos a partir de datos contenidos en listas en el lenguaje de programación Python. En este proyecto, para facilitar la interpretación de los resultados, se han visualizado, y esta ha sido la principal herramienta para hacerlo.

[5] https://pandas.pydata.org/
[6] https://numpy.org/
[7] https://scikit-learn.org
[8] https://matplotlib.org/

Seaborn

Seaborn [9] es otra librería de visualización de datos de Python basada en Matplotlib. Se ha utilizado para generar gráficos que no se puede generar con Matplotlib.

Yellowbrick

Yellowbrick [10], otro visualizador de datos de Python centrados en visualizar datos de Machine Learning. En este proyecto se ha utilizado para generar los gráfico del método del codo ya que cuenta con funciones automatizadas para ello.

kneed

`kneed` [11] es un repositorio de Python que implementa el método del codo de forma cuantitativa y que se ha utilizado en este proyecto para no aplicar el método del codo de forma visual, teniendo una base matemática para ello.

3.3.3. Xampp

XAMPP [12] es un paquete de software libre que incluye el sistema de gestión de bases de datos *MySQL*, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. Esta herramienta ha sido elegida porque tenemos experiencia previa gestionando bases de datos con *phpMyAdmin*, ya que la hemos utilizado durante el grado.

3.4. Formación

Durante el proceso de investigación se han encontrado cursos cuyos conocimientos ha incrementado la calidad del proyecto. La plataforma donde se encuentra este conocimiento es Kaggle [13]. En esta plataforma existen, de forma gratuita, varios cursos relacionados con el *Machine Learning*.

[9] https://seaborn.pydata.org/
[10] https://www.scikit-yb.org/en/latest/
[11] https://pypi.org/project/kneed/
[12] https://www.apachefriends.org/es/index.html
[13] https://www.kaggle.com/

Para este proyecto, se ha realizado el curso *Feature Engineering* donde se aprende a procesar los datos, estudiarlos, transformarlos, comprender el análisis *PCA* y realizar correctamente un análisis de grupos.

CAPÍTULO 4

Desarrollo del proyecto

En este apartado se describe el trabajo realizado en las diferentes fases de desarrollo del proyecto. Para ello, seguiremos el orden de la metodología KDD descrita en la sección 3.2 y presentada en la figura 3.2. La documentación del desarrollo esta compuesta por estas fases:

- Extracción de datos desde la base de datos (4.1)
- Preprocesamiento y limpieza (4.2)
- Estudio de las variables (4.3)
- *Clustering* (4.4)
- Análisis de la partición (4.5)

4.1. Extracción desde bases de datos

El desarrollo del proyecto parte de la base de datos de la plataforma Toolbox.Academy. Aquí se encuentra toda la información en diferentes tablas y el objetivo de esta fase consiste seleccionar los atributos que nos interesan para generar un *dataset* que represente adecuadamente al alumnado.

La organización responsable del desarrollo de la plataforma nos ha facilitado la base de datos en formato SQL. Para garantizar la privacidad de los usuarios los datos han sido previamente anonimizados. La base de datos ha sido consultada mediante un servidor local de *phpMyAdmin* lanzado desde Xampp. De esta forma se han podido hacer consultas a la base de datos desde una interfaz gráfica, viendo los resultados de una forma más visual e interactiva.

Respecto a la estructura, la base de datos cuenta con treinta tablas con distintos atributos. En este proyecto, nos hemos centrado en las tablas `student` y `logs`. Estas dos tablas se relacionan mediante la clave `user_id` de tal forma que cada fila de `logs` pertenece a un estudiante. La tabla `logs` recopila datos periódicos de la actividad del alumnado en la plataforma.

Teniendo en cuenta la estructura de la base de datos, mediante un comando SQL por un lado seleccionamos los atributos de ID de estudiante (*user_id*) y su fecha de nacimiento para calcular la edad (*age*) y por otro lado, haciendo el cómputo de todos los registros (*logs*) que tiene cada usuario en la plataforma y utilizando la agrupación por ID de usuario, se calcula la suma total del tiempo (*total_time*), aciertos (*total_ok*), errores (*total_error*), pistas solicitadas (*total_tip*) y consultas a los apuntes (*total_wiki*).

Una vez obtenidos los resultados, se han exportado en formato *csv* desde *phpMyAdmin* para poder leer los datos en Python.

A continuación, pasamos a tratar los datos con Python. Para ello, el archivo *csv* inicial es leído con Pandas y guardado como DataFrame. En la figura 4.1 se presenta el contenido del DataFrame que indica que el tamaño inicial es de 14170 filas x 7 columnas.

	user_id	age	total_ok	total_error	total_tip	total_wiki	total_time
0	117	15	202	181	11	7	15461
1	1009	14	448	387	25	12	43353
2	1010	14	284	302	0	1	40706
3	1011	14	462	274	8	0	36736
4	1012	14	306	511	19	7	35119
...	...	...	...	...	...	...	...
14165	32292	18	0	0	0	0	0
14166	32293	18	0	0	0	0	0
14167	32294	11	0	0	0	0	0
14168	32295	16	0	0	0	0	0
14169	32296	14	0	0	0	0	0

14170 rows × 7 columns

Figura 4.1: Estructura del *dataset* inicial

.

4.2. Preprocesamiento y limpieza

En esta fase analizaremos los datos iniciales con el objetivo de identificar inconsistencias y descartar casos que no nos interesen.

4.2.1. Datos inconsistentes

Consideraremos datos inconsistentes, valores nulos o incoherentes de los atributos y filas duplicadas. Primero, la función `info()` de la clase DataFrame de `pandas` nos ofrece la posibilidad de contabilizar por atributo los valores no nulos, con lo cual, teniendo el cuenta el tamaño inicial podemos identificar estos valores y eliminarlos directamente con la función `dropna()` de la clase DataFrame de `pandas`. El resultado de la función `info()`:

Como se aprecia en la figura 4.2, no hay valores nulos o filas duplicadas.

A continuación, uniremos la verificación de valores incoherentes con el descarte de casos no interesantes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14170 entries, 0 to 14169
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   user_id      14170 non-null  int64
 1   age          14170 non-null  int64
 2   total_ok     14170 non-null  int64
 3   total_error  14170 non-null  int64
 4   total_tip    14170 non-null  int64
 5   total_wiki   14170 non-null  int64
 6   total_time   14170 non-null  int64
dtypes: int64(7)
memory usage: 775.0 KB


Numero de filas duplicadas:  0
```

Figura 4.2: Resultado de la función `info()` del *dataset* inicial

4.2.2. Descartar casos

Otra de las funciones interesantes que ofrece la clase DataFrame de `pandas` es la función `describe()` que ofrece datos estadísticos de cada atributo como el mínimo, máximo, media, mediana y cuartillas, entre otros. Con esta función, podemos ver en qué rangos se encuentran los atributos y descartar casos incoherentes o que se adecuen con el objetivo de este proyecto.

	user_id	age	total_ok	total_error	total_tip	total_wiki	total_time
count	14170.000000	14170.000000	14170.000000	14170.000000	14170.000000	14170.000000	1.417000e+04
mean	16361.786733	16.768596	258.828652	186.211150	3.342696	2.304940	3.300159e+04
std	9339.131026	62.699426	249.632280	336.355066	11.770897	10.777723	1.577240e+05
min	117.000000	-7.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00
25%	8303.250000	13.000000	72.000000	10.000000	0.000000	0.000000	2.146250e+03
50%	15967.500000	15.000000	211.000000	58.000000	0.000000	0.000000	8.610000e+03
75%	24297.750000	17.000000	365.000000	206.000000	2.000000	0.000000	2.394425e+04
max	32296.000000	2021.000000	2589.000000	5797.000000	378.000000	386.000000	8.228371e+06

Figura 4.3: Resultado de la función `describe()` del *dataset* inicial

Analizando la figura 4.3, podemos identificar incoherencias como por ejemplo, edades

negativas. La omisión de estos casos se ha hecho a la vez que la selección de estudiantes entre 6 y 18 años, es decir, $age \in [6, 18]$. En esta criba se han descartado 1378 casos.

Analizando el atributo *total_time* podemos apreciar que existen usuarios que no han tenido ninguna actividad en la plataforma, es decir, no nos van a aportar ninguna información válida. Es por ello que se establece un mínimo de diez minutos de actividad, teniendo en cuenta que estamos tratando segundos, $total\_time \geq 600$.

En total, en esta criba se han descartado 1645 casos. Tras el preprocesamiento y limpieza, se han descartado en total 3023 casos y el segundo *dataset* ha pasado a tener 11147 filas x 7 columnas y estas estadísticas (ver figura 4.4):

	user_id	age	total_ok	total_error	total_tip	total_wiki	total_time
count	11147.000000	11147.000000	11147.000000	11147.000000	11147.000000	11147.000000	1.114700e+04
mean	16537.237553	14.131156	295.624922	218.240244	3.850632	2.659101	3.874490e+04
std	9233.725152	2.640732	246.019733	360.098135	12.851602	11.151191	1.753336e+05
min	117.000000	6.000000	0.000000	0.000000	0.000000	0.000000	6.050000e+02
25%	8831.500000	12.000000	117.000000	23.000000	0.000000	0.000000	3.944500e+03
50%	15999.000000	14.000000	249.000000	84.000000	0.000000	0.000000	1.131200e+04
75%	24190.500000	16.000000	394.500000	253.000000	2.000000	0.000000	2.947900e+04
max	32284.000000	18.000000	2589.000000	5797.000000	378.000000	386.000000	8.228371e+06

Figura 4.4: Resultado de la función `describe()` del *dataset* post-procesamiento

4.3. Estudio de las variables

Una vez tengamos los datos procesados y limpiados, pasaremos a analizarlos. Para ello, visualizaremos y haremos un análisis de componentes principales o *Principal Component Analysis* (PCA) como técnica descriptiva y en un segundo paso, haremos una posible reducción de dimensionalidad. Sin embargo, el análisis PCA es sensible a los valores extremos con lo cual empezaremos esta fase con escalar y transformar los datos. Por último, estudiaremos la creación de nuevas variables combinándolas y creando ratios.

4.3.1. Primer vistazo a los datos

En este paso el objetivo consiste en visualizar los datos para identificar qué tipo de distribución siguen los datos, si siguen alguno, e identificar la mejor transformación para

adaptarlos a una distribución normal. Los algoritmos de aprendizaje, tanto supervisado como no supervisado, funcionan mejor cuando los datos siguen una distribución normal.

La figura 4.5 visualiza la distribución y forma que tienen los datos iniciales después del procesamiento.

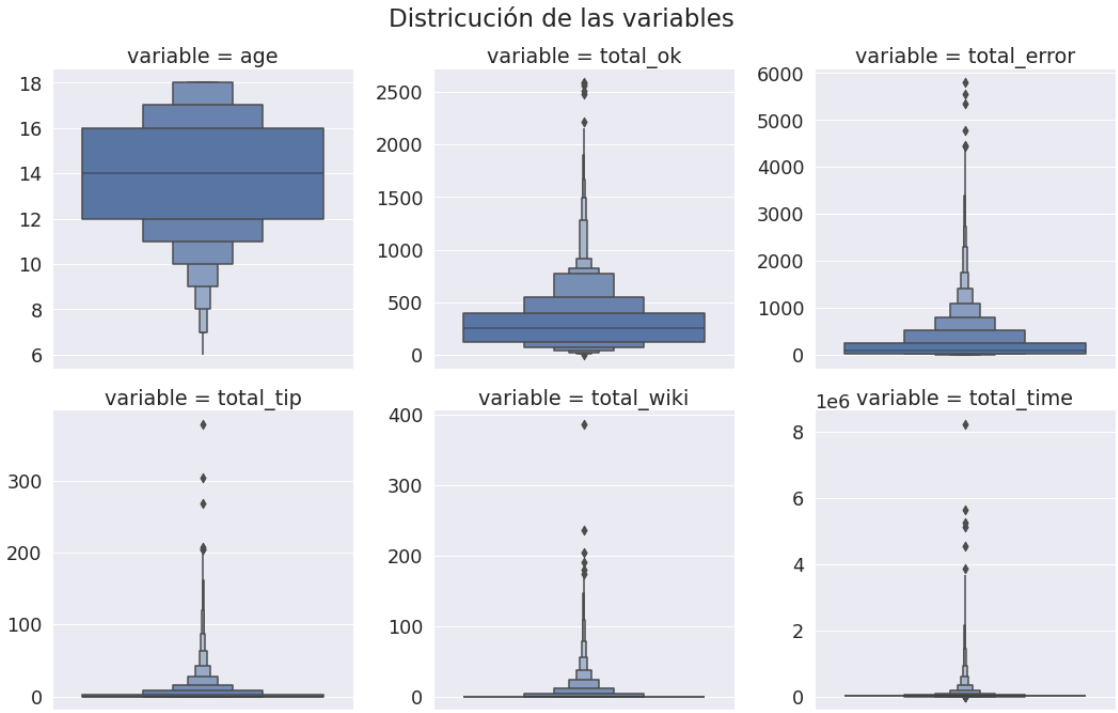

Figura 4.5: Visualización de la distribución de los datos preprocesados

Analizando los gráficos se aprecian varias características relevantes. Primero, no todos los datos tienen la misma escala. La variable `age` esta entre 6 y 18, como lo hemos filtrado en el preprocesamiento, y `total_ok` esta entre 0 y 2500. Segundo, las variables no tienen una distribución adecuada, ya que se agrupan en un estrecho intervalo. Es por ello que necesitan ser transformados.

4.3.2. Transformaciones

Una vez se haya identificado la necesidad de transformar los datos, vamos a aplicar a cada variable la transformación más adecuada según su distribución. Son tres la transformaciones estudiadas [Google, 2021]: escalado, transformación logarítmica y transformación de cuartillas.

- **Escalado**: mediante esta transformación se consigue que los datos estén en el mismo rango. Se aplica a los datos que siguen una distribución Gausiana.

- **Transformación logarítmica**: esta transformación consiste en aplicar una función logarítmica a los datos. Se aplica cuando los datos siguen una distribución exponencial.

- **Transformación de cuartillas**: esta transformación consiste en dividir los datos en intervalos con el mismo número de casos, reemplazar los datos por el índice de su intervalo y finalmente, escalar los índices. Esta transformación se aplica cuando los datos no siguen una distribución especifica como campana de Gauss o ley exponencial.

Una vez definidas las transformaciones, se ha analizado cada variable para elegir la transformación más adecuada.

Age

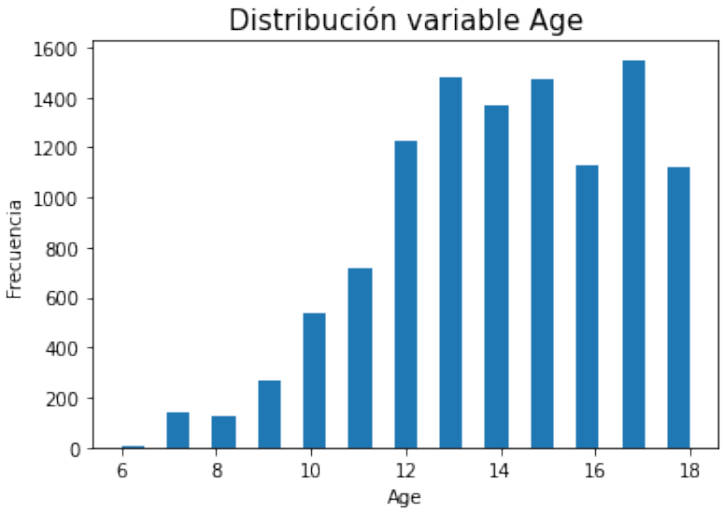

Figura 4.6: Visualización de la distribución de la variable age

Como se aprecia en la figura 4.6, esta variable se parece a una distribución Gaussiana, aunque no es exacta. Ante esta forma, aplicaremos la transformación por cuartillas para ver si se acerca o no a la forma de campana de Gauss.

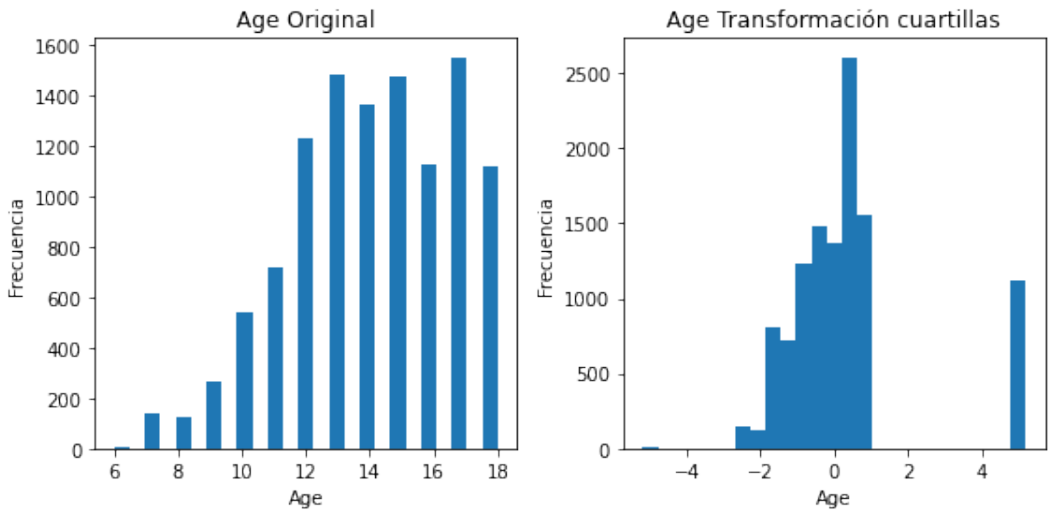

Figura 4.7: Comparación de la distribución de la variable *age* original con transformación de cuartillas

Comparando los gráficos (ver figura 4.7) se observa que los datos originales tienen una mejor distribución, con lo cual, no aplicaremos ninguna transformación en la variable *age*

Total_ok

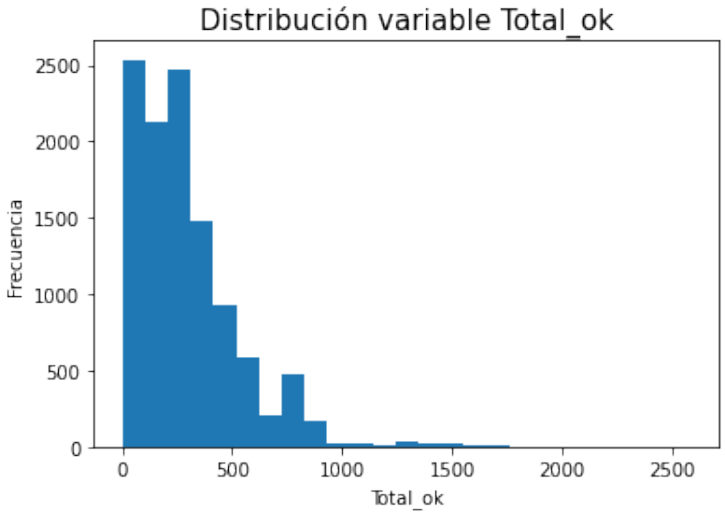

Figura 4.8: Visualización de la distribución de la variable total_ok

Como se aprecia en la figura 4.8 , la variable *total_ok* sigue una distribución exponencial, por lo que se ha aplicado una transformación logarítmica. En este caso, encontramos que

la variable puede tomar valor cero, con lo cual la función de transformación tiene que ser $\log(x+1)$. Para ello, utilizamos la función `log1p()` del paquete `Numpy` de Python.

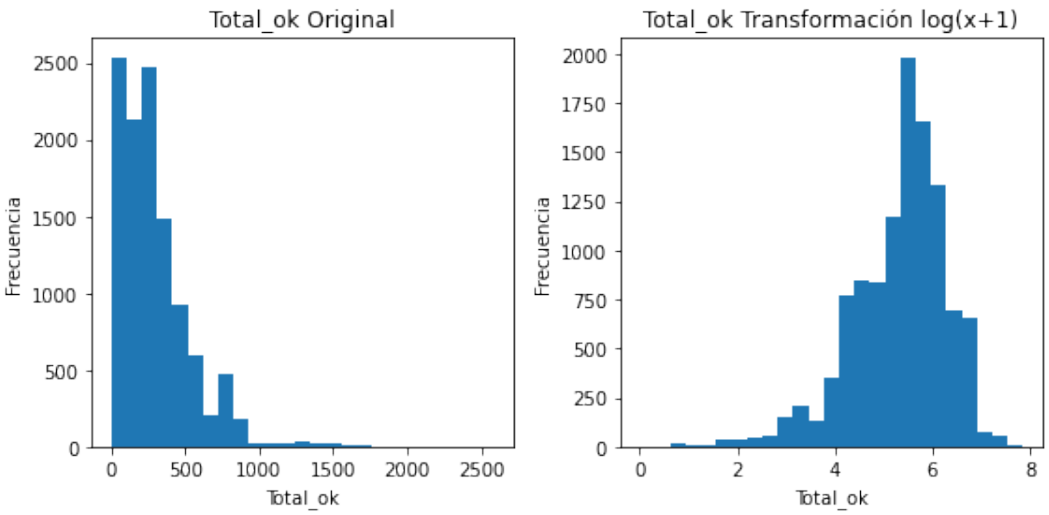

Figura 4.9: Comparación de la distribución de la variable *total_ok* original y con transformación logarítmica

En la comparación (ver figura 4.9) se aprecia claramente la mejora en cuanto a la distribución, con lo cual, utilizaremos los datos transformados de aquí en adelante.

Total_error

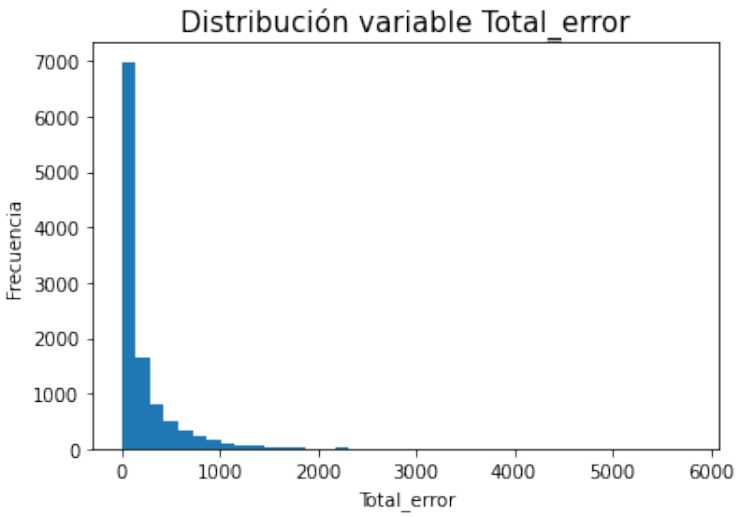

Figura 4.10: Visualización de la distribución de la variable total_error

La variable *total_error* también sigue una distribución exponencial (ver figura 4.10). Es por ello que también le aplicaremos una transformación logarítmica. Al igual que con la variable *total_ok* puede tomar el valor cero, por lo que la función de transformación debe ser $\log(x+1)$.

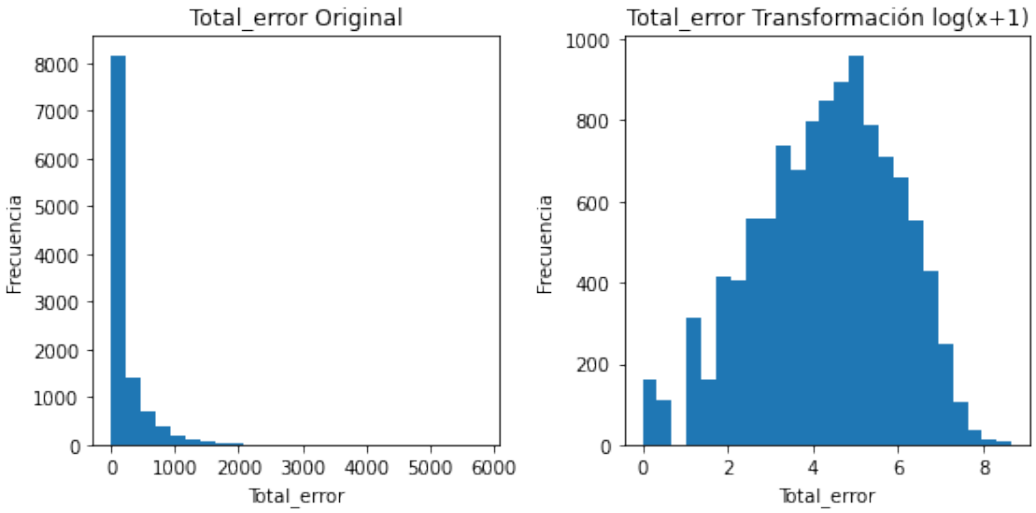

Figura 4.11: Comparación entre la distribución de la variable *total_error* original y con transformación logarítmica

La mejora en la distribución es clara (ver figura 4.11). Los datos transformados de la variable *total_error* serán los utilizados, en adelante.

Total_tip

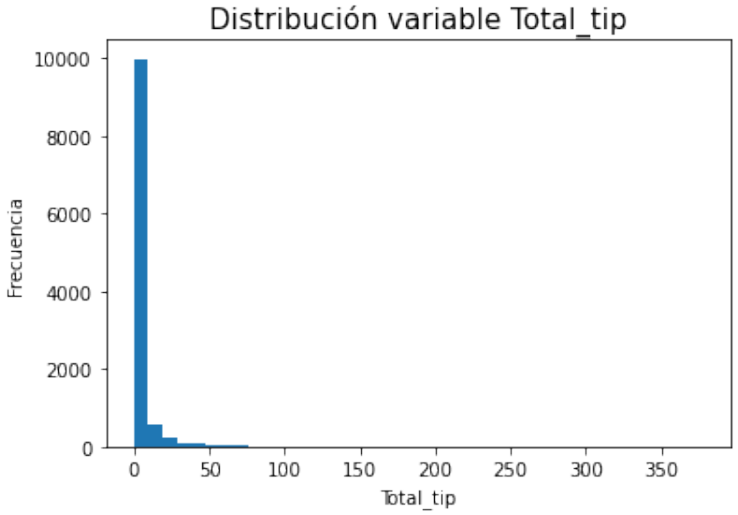

Figura 4.12: Visualización de la distribución de la variable total_tip

La distribución de esta variable (ver figura 4.12) se parece a una exponencial, sin embargo, los casos se agrupan demasiado en torno al cero e identificamos casos extremos en la figura 4.5. Dicho esto, estaríamos ante una distribución no exponencial y no Gaussiana, con lo cual la mejor transformación es la de cuartillas.

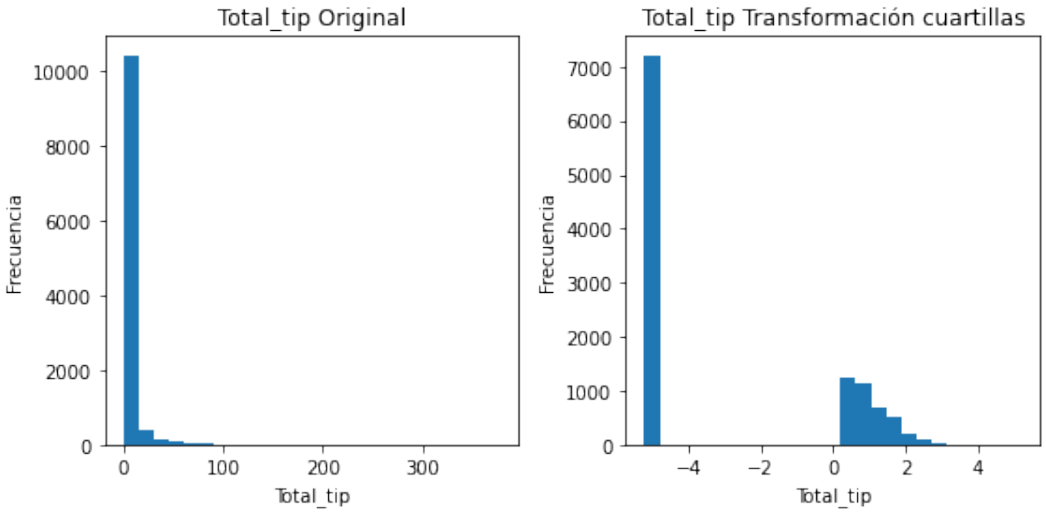

Figura 4.13: Comparación entre la distribución de la variable *total_tip* original y con transformación de cuartillas

Analizando las dos distribuciones (ver figura 4.13), la transformación de cuartillas ha hecho que los casos cercanos al cero o cero se agrupen y se desglosen los demás casos

pronunciando su distanciamiento respecto a los cercanos al cero. Por ello, utilizaremos los datos transformados.

Total_wiki

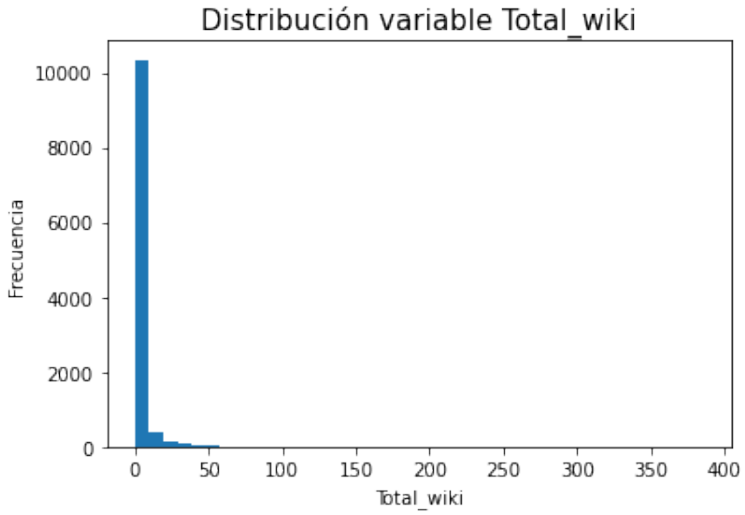

Figura 4.14: Visualización de la distribución de la variable total_wiki

Al igual que con la variable *total_tip*, parece una distribución exponencial (ver figura 4.14), pero encontramos la mayoría de casos agrupados en el cero y varios casos extremos como muestra la figura 4.5. Es por ello que se opta por la transformación de cuartillas. Este cambio se aprecia en la figura 4.15

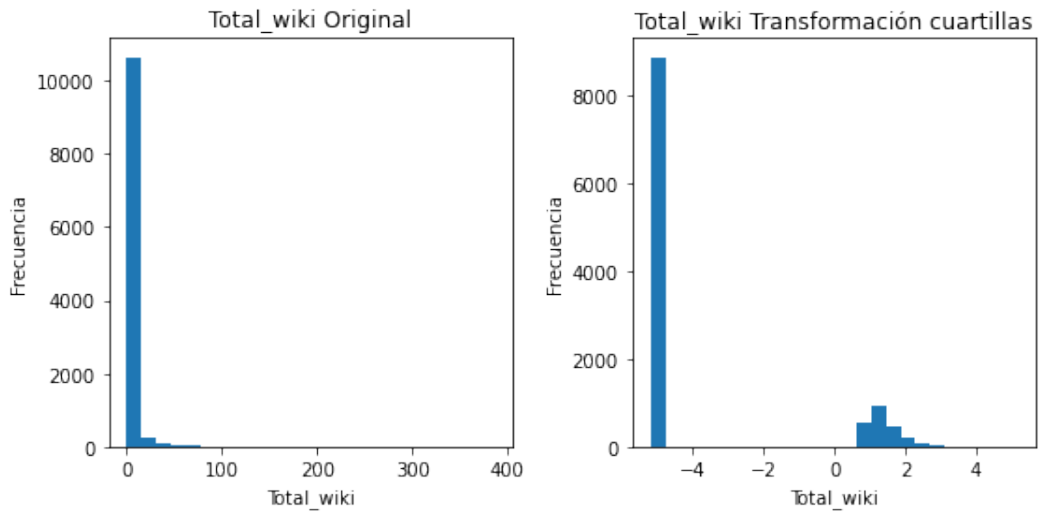

Figura 4.15: Comparación entre la distribución de la variable *total_wiki* original y con transformación de cuartillas

Al igual que con la variable *total_tip*, la transformación de cuartillas ha hecho que los casos cercanos al cero se agrupe y que se pronuncie la diferencia respecto a los demás casos.

Total_time

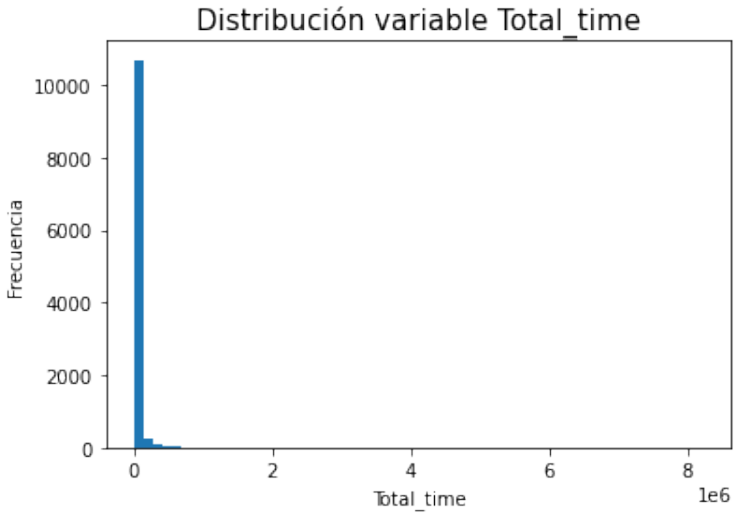

Figura 4.16: Visualización de la distribución de la variable total_time

Analizando la distribución de la variable que mide el tiempo (ver figura 4.16), se aprecia la misma característica que en las variables *total_tip* y *total_wiki*, con lo cual, la mejor

transformación es la de cuartillas. En la figura 4.17 se aprecia el cambio de los datos originales a los transformados con la transformación de cuartillas.

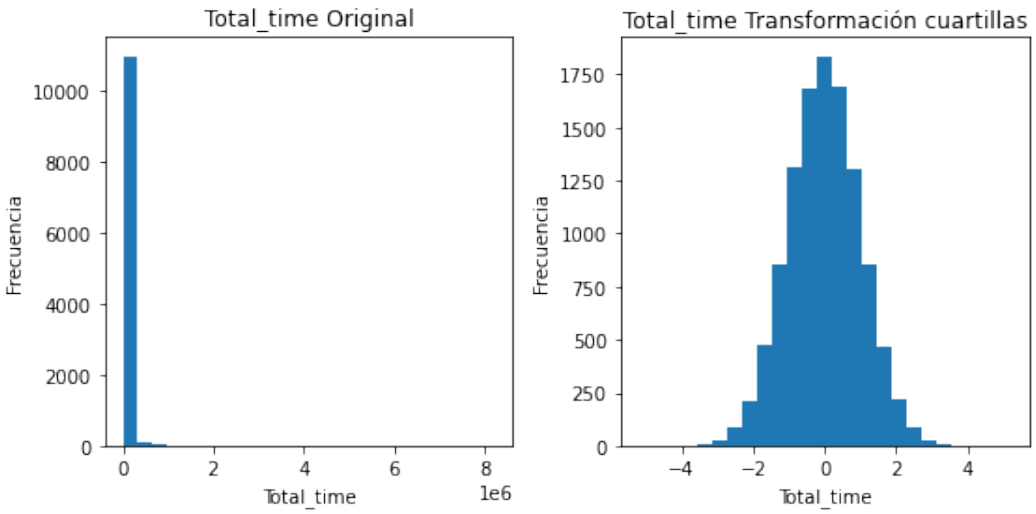

Figura 4.17: Comparación entre la distribución de la variable *total_time* original y con transformación de cuartillas

A modo de resumen, en la tabla ?? se presentan las tranformaciones que se ha aplicado a cada variable:

Atributo	Transformación
age	-
total_ok	$\log(1+x)$
total_error	$\log(1+x)$
total_tip	*Cuartillas*
total_wiki	*Cuartillas*
total_time	*Cuartillas*

Tabla 4.1: Transformación aplicada a cada variable.

Para finalizar la fase de transformación, cabe añadir que no todas las variables están en la misma magnitud y que la técnica PCA es sensible a valores extremos. Con lo cual, es necesario escalar los datos antes de aplicarla. El proceso de escalado consiste en el siguiente cálculo, donde μ es la media y σ la desviación de la variable X.

$$z = \frac{(X-\mu)}{\sigma}$$

Al estar trabajando con Python, contamos con varias librerías que ofrecen funciones para escalar los datos. En el proyecto se ha utilizado la clase `StandarScaler`[1] de la librería `sklearn`. En la figura 4.18 visualizamos la distribución del *dataset* transformado y escalado:

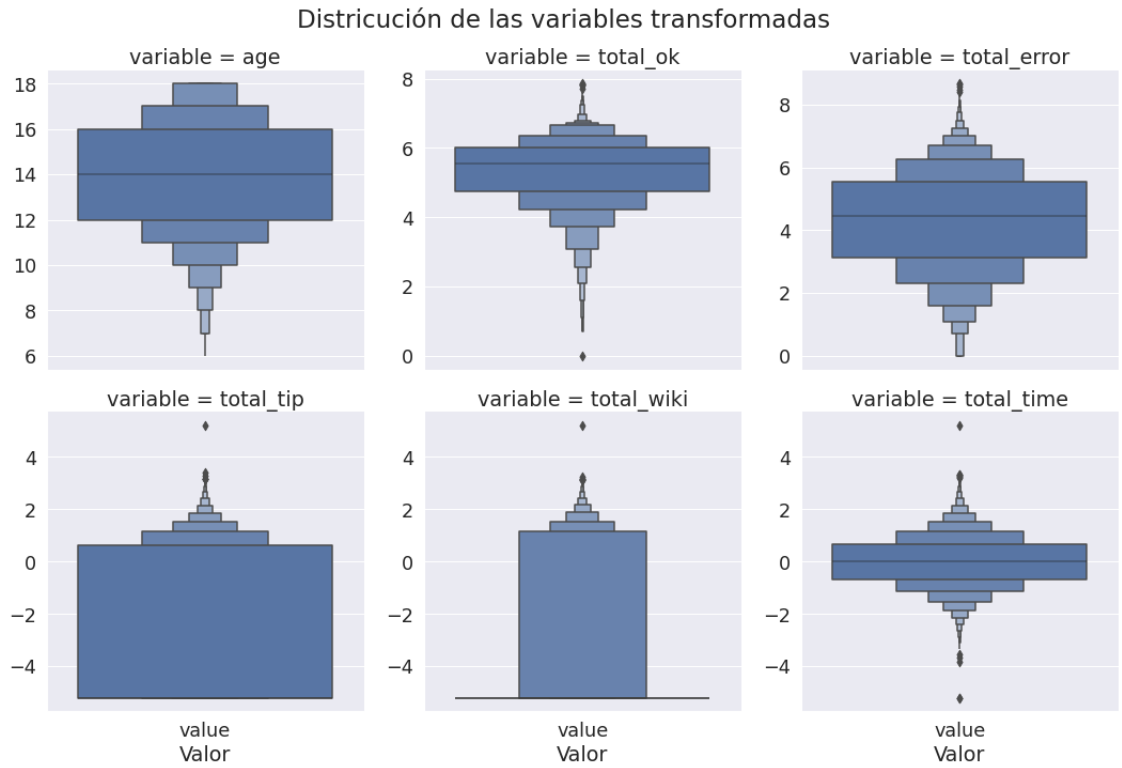

Figura 4.18: Distribución final de todas las variables tras la transformación

4.3.3. Análisis de componentes principales (PCA)

El análisis de componentes principales o PCA es una técnica [Abdi and Williams, 2010] utilizada para describir un conjunto de datos en términos de nuevas variables o componentes no correlacionadas. Se basa en el cálculo de la descomposición de autovalores de la matriz de covarianza. El objetivo aplicar esta técnica es analizar cuanta varianza captura cada variable.

Al estar trabajando en Python, existe dentro de la librería `sklearn.decomposition` una

[1] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

función llamada `PCA` [2] que permite hacer este análisis. Aplicando la función al *dataset* obtenido en la fase anterior obtenemos el resultado mostrado en la tabla 4.2

	PC1	PC2	PC3	PC4	PC5	PC6
age	-0.081493	-0.889018	0.434300	0.097580	0.045581	0.052819
total_ok	-0.511458	-0.126961	-0.207404	-0.222103	-0.782042	-0.135484
total_error	-0.534537	-0.027412	-0.109069	-0.107511	0.525025	-0.643749
total_tip	-0.333605	0.174584	0.097120	0.915731	-0.099316	0.019186
total_wiki	-0.235260	0.402857	0.842647	-0.259531	-0.065677	0.025204
total_time	-0.528583	-0.001857	-0.192312	-0.153850	0.310652	0.750625

Tabla 4.2: Transformación aplicada a cada variable.

Junto a los componentes, los gráficos de varianza de cada componente y varianza acumulada han sido de ayuda para sacar conclusiones del análisis PCA (ver figura 4.19).

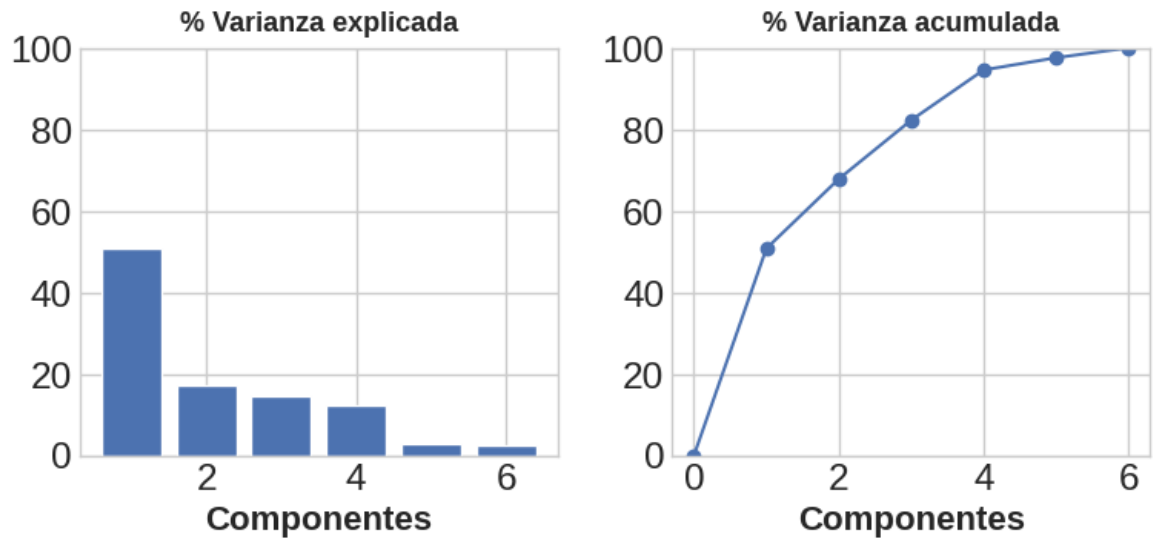

Figura 4.19: Gráfico del porcentaje de varianza explicada y varianza acumulada por componente principal

Basándonos en la información que hemos obtenido, por un lado sacaremos conclusiones de los valores de los componentes principales y luego, estudiaremos la posibilidad de crear nuevas variables, combinando las actuales, mediante ratios.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Con los cuatro primeros componentes principales prácticamente se representa la totalidad de la varianza. Analizando los valores de cada componente principal, podemos identificar que el primero (`PC1`) se basa en las variables `total_ok`, `total_error` y `total_time`. Los componentes principales dos (`PC2`) y tres (`PC3`) vienen a transmitir que cuanto menor sea la edad, más se accede a los apuntes y a las pistas. Por último, el componente principal cuatro (PC4) se basa principalmente en la variable `total_tip`.

Además, teniendo el cuenta el porcentaje de varianza que representa PC1 y las variables en las que se basa identificamos la necesidad de crear nuevas variables mediante ratios. Bajo la lógica del rendimiento académico pueden podemos crear la variable precisión. La precisión (`accuracy`) que tiene cada usuario la podemos representar mediante el siguiente ratio:

$$accuracy = \frac{total\_ok}{total\_ok + total\_error}$$

Ante esta nueva variable encontramos una casuística que hay evitar, cuando $total\_ok + total\_error$ es cero. Realizando una búsqueda con Python en el DataFrame que tenemos guardado nuestro *dataset* no tenemos ningún caso con estas características, con lo cual no tendremos que actualizar el *dataset*.

Como conclusión del PCA podemos afirmar lo siguiente. Primero, hemos identificado las variables más representativas, es decir, las más involucradas en representar la varianza en los componentes principales(`total_ok`, `total_error` o `accuracy` y `total_time`). Segundo, hemos identificado una nueva variable (`accuracy`) que mediante el ratio de las variables `total_ok` y `total_error` ayuda en la medición de rendimiento académico. De esta forma, pasaremos a la siguiente fase con un *datasets* con los atributos `accuracy`, `total_tip`, `total_wiki` y `total_time`.

4.4. *Clustering*

En esta fase se procede a hacer el análisis de grupos o *clustering* de los datos obtenidos en la fase anterior. Primero, se decide el número de grupos óptimo, en segundo lugar, probaremos tres tipos de algoritmos para agrupar y evaluaremos los resultados. Por último, la mejor configuración para el *clustering* se guardará para analizar los grupos en la última fase de este proyecto.

4.4.1. Número de grupos

Para iniciar el análisis de grupos, al tratar con datos no etiquetados, se tiene que estudiar los datos para introducir en el algoritmo de aprendizaje no supervisado encargado de hacer la agrupación, el número de grupos. Para ello, no existe un criterio objetivo, sin embargo, existen varios métodos e índices. En este proyecto vamos a utilizar dos, el método del codo y el índice Silhouette.

Antes de empezar, cabe recalcar que la metodología para elegir el número de *clusters* será visualizando las métricas en un gráfico donde en el eje horizontal están los diferentes número de grupos o particiones y en el eje vertical el índice.

Método del codo

El método del codo (*Elbow Method* [Ketchen and Shook, 1996]) es un heurístico usado para determinar el número óptimo de *clusters* en un *dataset* que establece como mejor resultado el punto que se identificaría en el gráfico como un codo. Para visualizar se utilizan en el eje horizontal los números de grupos que para cada cual se aplica el algoritmo K-Means. A partir del resultado del algoritmo, para cada partición se calcula la distorsión y se representa en el eje vertical.

El índice de distorsión es la suma de las distancias al cuadrado desde cada punto hasta su *centroide* asignado:

$$J(c,\mu) = \sum_{i=1}^{N} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

donde μ es el *centroide*.

Para aplicar este heurístico, se ha utilizado la librería de Python `Yellowbrick` que en su paquete de `cluster` incluye la función `KElbowVisualizer()` para generar el gráfico del método del codo en el rango de número de grupos especificado.

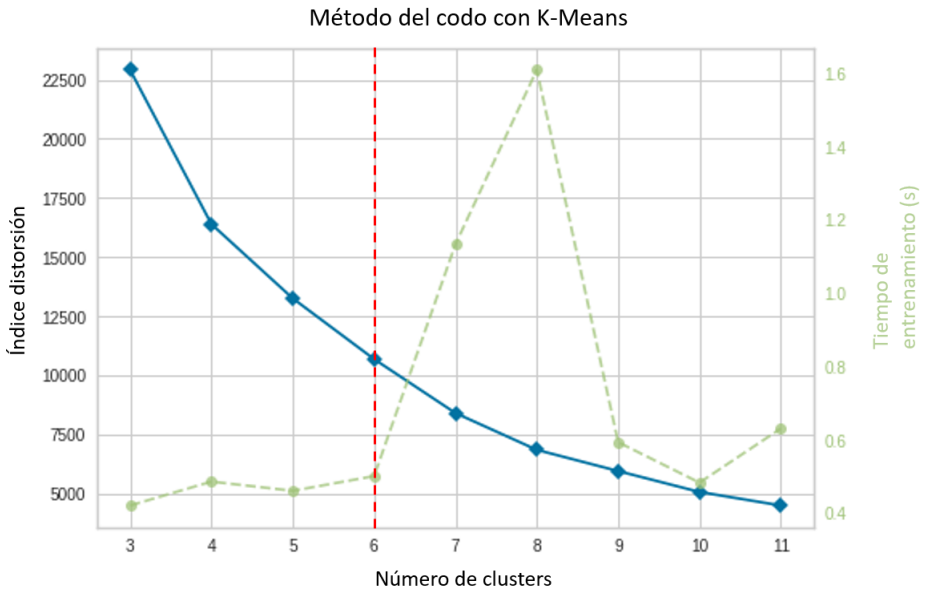

Figura 4.20: Gráfico método del codo realizado con K-Means.

Según el resultado de la figura 4.20 este método el número óptimo de *clusters* para el *dataset* es seis ($k = 6$). Una vez visto este resultado, se ha contrastado con el segundo criterio, el gráfico de índices Silhouette.

Índice Silhouette

El segundo criterio consiste en calcular el índice Silhouette de las particiones con diferentes números de grupos. El índice Silhouette [Rousseeuw, 1987] es una medida de cuán similar es un objeto a su propio cúmulo (cohesión) en comparación con otros cúmulos (separación). El resultado es un coeficiente entre -1 y 1. El *clustering*, como en el método del codo, se ha hecho con el algoritmo K-Means aunque en el siguiente paso estudiaremos otros algoritmos.

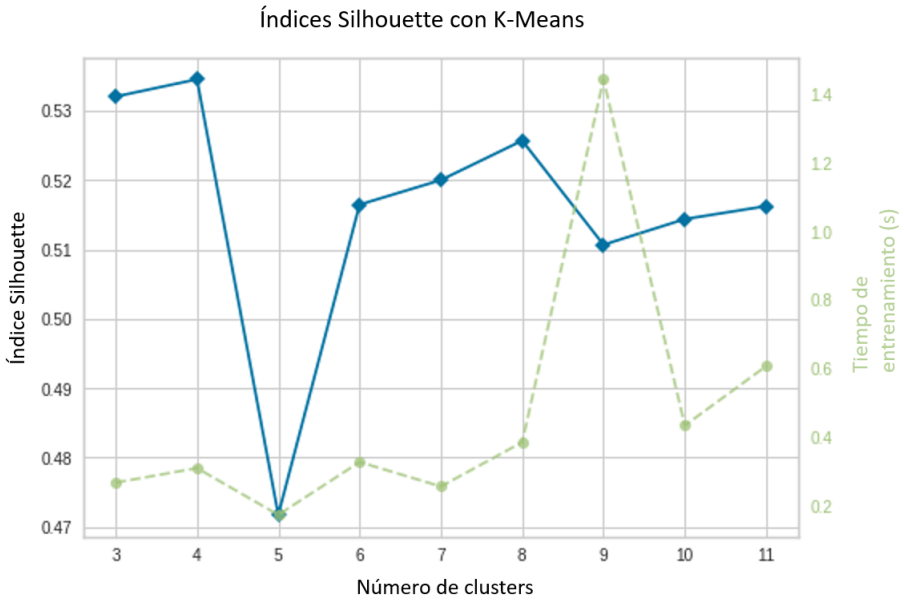

Figura 4.21: Índices Silhouette para diferentes particiones realizados con K-Means

Según la figura 4.21 la mejor partición es cuando se separan los datos en cuatro grupos.

Conclusión

Según el método del codo el número óptimo de *clusters* es seis y según el índice Silhouette es de cuatro. Cabe recalcar, que el índice Silhouette para seis clusters solo se reduce un $1,8\,\%$, con lo cual, el número de grupos óptimo para nuestro *dataset* es de seis.

4.4.2. Algoritmos de agrupación

Una vez establecido el número de grupos, podemos introducirlo como parámetro en diferentes algoritmos de agrupación. Existen varios algoritmos para abordar la tarea de agrupar [Xu and Tian, 2015] según la distribución de cada *dataset*. Principalmente se conocen tres tipos de algoritmos para esta tarea: basados en *centroides* o en distancia, basados en densidad y jerárquicos.

Algoritmos basados en distancia

Los algoritmos basados en *centroides* o basados en distancias elijen al azar k elementos o *centroides* (donde k es un parámetro de entrada) y asignan a los demás elementos un grupo $[1,k]$ en función de la distancia media que tengan a cada grupo. El algoritmo de este tipo más utilizado es K-Means y es el utilizado para este trabajo.

El algoritmo K-Means [MacQueen, 1967] funciona dado un conjunto de observaciones $(x_1, x_2, ..., X_n)$ cada una de d dimensiones, crea una partición de k grupos ($S = \{S_1, S_2, ..., S_k\}$) en el que cada caso pertenece al grupo cuyo valor medio es más cercano, es decir,

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

donde μ_i es la media de los puntos en S_i.

Para aplicar este algoritmo se ha utilizado una implementación de la librería de Python `sklearn`. En el paquete `cluster` de esta librería se encuentra la función `KMeans()` que implementa el algoritmo. Se ha añadido como parámetro de entrada el numero de *clusters* con el valor 6 (establecido en el apartado 4.4.1). El problema a resolver es NP-Hard, sin embargo, si la dimensión (d) y el número de *clusters* (k) son fijados tiene la orden de $\Theta(n^{dk+1})$. Además, si se utilizan algoritmos heurísticos se puede reducir su complejidad. En la implementación utilizada, mediante el algoritmo heurístico Lloyd's [Lloyd, 1982] se obtiene una complejidad de $O(knT)$, donde n es el número de casos y T el número de iteraciones.

Una vez aplicado el algoritmo, se evalúa calculando el índice Silhouette de la partición. En este caso, ya se ha calculado para crear el gráfico 4.21. Sin embargo, se puede calcular este índice utilizando utilizando la función `silhouette_score` del paquete `sklearn.metrics`. El índice Silhouette de la partición realizada con K-Means para seis grupos es 0.4502.

Para facilitar la comprensión de los resultados se han visualizado. Por un lado se ha creado un gráfico de tres dimensiones para evitar, en la medida de lo posible, la superposición de puntos. Para ello se ha hecho una reducción de la dimensionalidad a tres componentes. En el gráfico de la figura 4.22 queda constancia del porcentaje de variabilidad que representa cada eje. Por otro lado, se han creado tres gráficos en los que se representan los tres componentes por parejas (ver figura 4.23).

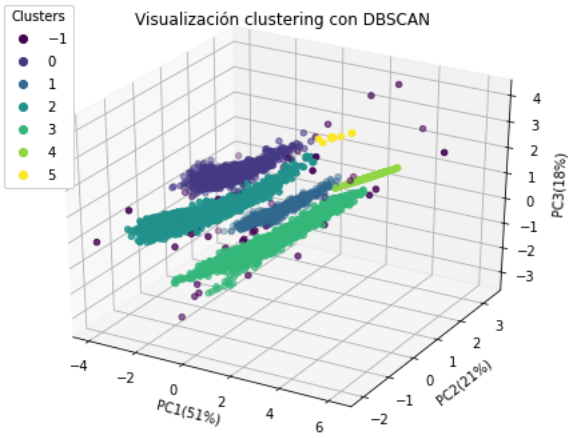

Figura 4.22: Visualización en tres dimensiones del clustering con K-Means

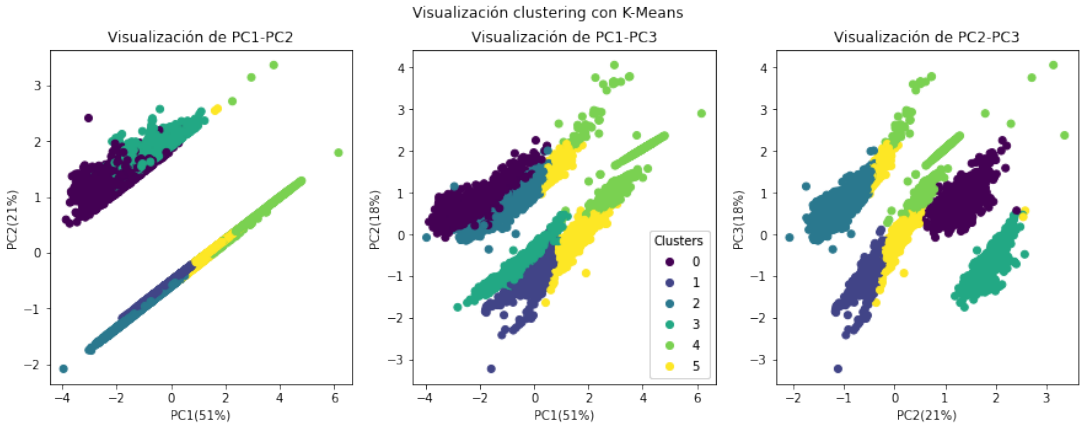

Figura 4.23: Visualización clustering con K-Means en dos dimensiones

Algoritmos basados en densidad

Este tipo de algoritmos tienen como objetivo detectar en qué áreas existe mayor concentración de puntos y en que áreas los puntos están más separados por vacíos o por poca densidad. Los puntos cercanos en áreas densas se etiquetan como un grupo y los puntos lejanos o que se encuentra en áreas de poca densidad se etiquetan como ruido. Este tipo de algoritmo no tienen como parámetro de entrada el número de grupos, ellos deciden el número de grupos según la distribución de los datos.

Para este proyecto se ha elegido el algoritmo DBSCAN [Ester et al., 1996]. Este algoritmo

es un algoritmo de agrupación basado en la densidad de la distribución de los datos. Requiere de dos parámetros de entrada:

- *minPuntos* : número mínimo de puntos para que una área se considera densa. Como regla de oro se establece $minPuntos = 2 \times d$ donde d es el número de dimensiones del *dataset*. En nuestro caso $minPuntos = 8$

- ε : distancia máxima entre puntos para que se consideren cercanos. Para decidir el valor de este parámetro se utiliza la siguiente regla de oro. Se calcula el punto $minPuntos - 1$ más cercano de cada caso. Una vez tengamos este dato de cada punto, se ordena en orden ascendente y se visualiza en un gráfico donde en el eje horizontal están los puntos y en el vertical las distancias. El valor de ε será elegido mediante el método del codo, es decir, el punto que visualmente es el codo del gráfico.

  Para ello, primero realizaremos el grafo de vecinos más cercanos utilizando la función de Python `NearestNeighbors()` del paquete `sklearn.neighbors`. Como parámetro de entrada se introduce el número de vecinos más cercanos, en nuestro caso, como queremos el $minPuntos - 1$, le daremos el valor 7, teniendo en cuenta que $minPuntos = 8$. A continuación, a partir del resultado, cogeremos la distancia del séptimo punto más cercano de cada punto, los ordenaremos para visualizarlos y aplicaremos el método del codo (ver Figura 4.24). Para aplicarlo, se ha utilizado la función `KneeLocator` de Python de la librería `kneed`.

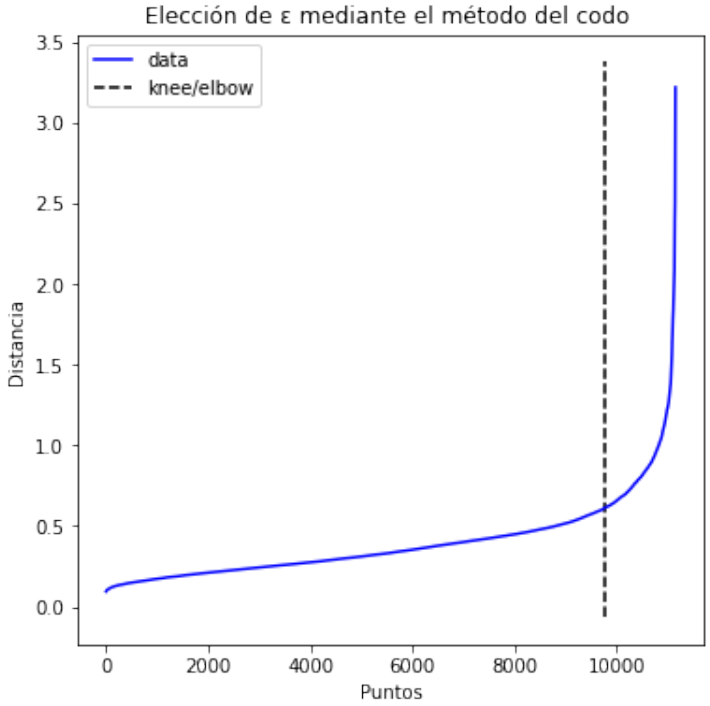

Figura 4.24: Elección de ε mediante el método del codo.

El punto que esta identificando en el gráfico de la Figura 4.24 indica el valor 0,6149, con lo cual, este será el valor de ε

Parámetro de entrada	Valor
ε	0,6149
minPuntos	8

Tabla 4.3: Parámetros de entrada para el algortimo DBSCAN

El algoritmo inicia por un punto arbitrario que no haya sido visto. Recoge todos los puntos no visitados que estén a una distancia menor o igual a ε y si hay más puntos que *minPuntos* se considera un grupo. De lo contrario, el punto es etiquetado como ruido. Respecto a la complejidad, el algoritmo es de complejidad $O(nd)$, ya que tiene que pasar por todos los puntos (n). Por otro lado, se tiene en cuenta el número de vecinos promedio de cada punto que están a una distancia ε (d). La implementación de este algoritmo viene en el paquete de Python `sklearn.cluster` en la función `DBSCAN()`.

Los resultados de este algoritmo se recogen en la Tabla 4.4

Cluster	Tamaño
1	1225
2	1058
3	2685
4	5978
5	144
6	10
-1	47

Tabla 4.4: Resultado análisis de grupos con DBSCAN

El algoritmo ha identificado seis clusters, la misma cantidad establecida en la fase 4.4.1. Además, etiqueta con -1 los casos de ruido.

El índice Silhouette para esta partición ha dado un valor de 0,4773. Como en el caso anterior, se ha creado un gráfico en tres dimensiones para evitar la superposición de puntos (ver Figura 4.25) y otro gráfico en dos dimensiones combinando los ejes por parejas (ver Figura 4.26).

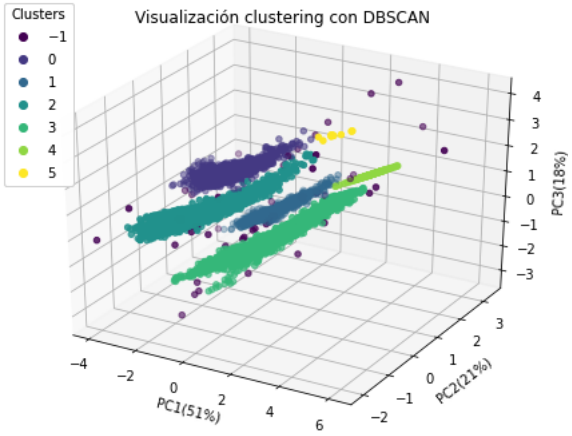

Figura 4.25: Visualización del *clustering* con DBSCAN en tres dimensiones

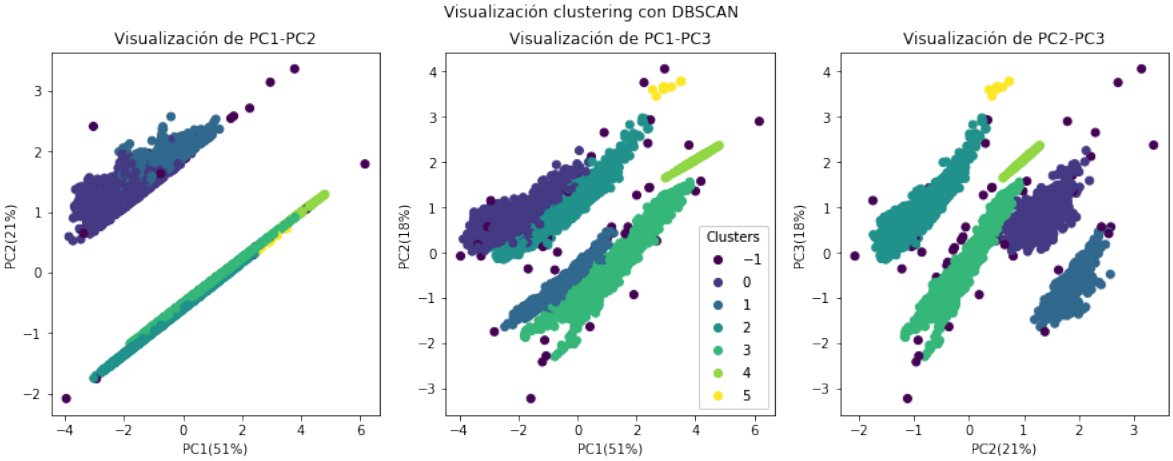

Figura 4.26: Visualización del *clustering* con DBSCAN en dos dimensiones

Jerárquicos

Los algoritmos jerárquicos [Johnson, 1967] son un tipo de método de análisis de grupos que buscan construir una jerarquía de grupos. Generalmente, hay dos tipos de estrategias, aglomerativas y divisivas. La primera empieza por abajo la estructura jerárquica, es decir, al comienzo cada caso es un grupo. Al contrario, la segunda, empieza desde arriba, es decir, todos los casos están en el mismo grupo y a medida que se construye la jerarquía hacia abajo se van dividiendo. Lo que tienen en común es que las dos estrategias se abordar con un algoritmo heurístico *greedy* ya que su complejidad, en términos generales, es de $O(n^3)$, lo cual es muy costoso para grandes volúmenes de datos. En este proyecto se ha utilizado la función `AgglomerativeClustering()` del paquete de Python `sklearn.cluster` para aplicarlo, obteniendo los resultados mostrados en las Figuras 4.27 y 4.28.

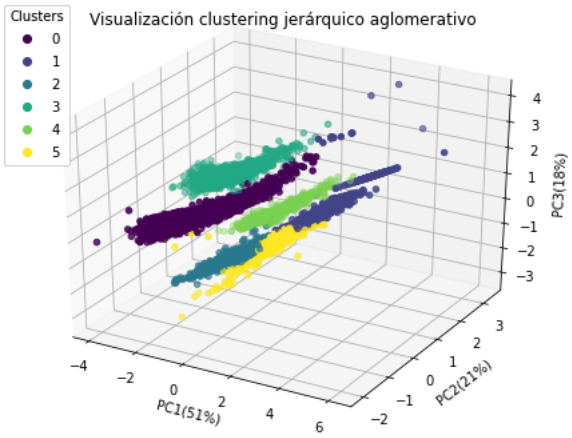

Figura 4.27: Visualización del *clustering* jerárquico en tres dimensiones

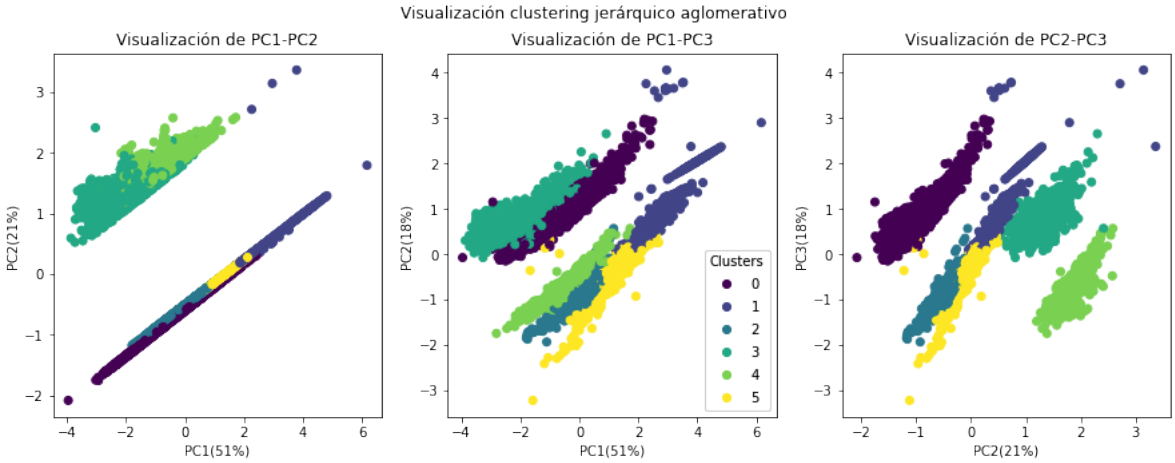

Figura 4.28: Visualización del *clustering* jerárquico en dos dimensiones

El índice Silhouette para esta partición ha sido de 0,418.

Conlusión

La Tabla 4.5 recoge los valores que ha cogido el índice Silhouette utilizado para evaluar los tres tipos de algoritmos aplicados en los datos.

Algoritmo	Valor Silhouette
K-Means	0,4502
DBSCAN	0,4773
Aglomerativo	0,41

Tabla 4.5: Resumen de los valores del índice Silhouette por tipo de algoritmo

En base a los resultados obtenidos deducimos las siguiente conclusiones. Primero, queda descartado el algoritmo jerárquico, ya que ha obtenido el peor resultado. Además es el más costoso, con lo cual, es el menos escalable ante una mayor adopción de datos. Dicho esto, los algoritmos K-Means y DBSCAN han obtenido resultados parecidos, aunque DBSCAN es ligeramente mejor. Ante esta situación, si tenemos en cuenta el coste del algoritmo son del misma orden, con lo cual, se ha optado por DBSCAN para hacer la partición de los datos.

4.5. Análisis de la particion o *profiling*

En esta última fase del proyecto se analiza que tipo de usuario se encuentra en cada grupo y se destaca el grupo con menor rendimiento académico o el grupo que pueda tener problemas en el aprendizaje, atendiendo a los objetivos definidos para el proyecto.

Para ello, se calculan varios valores estadísticos descriptivos de cada *cluster*. Cabe añadir que como ya se dispone de las etiquetas de grupo, utilizaremos los datos originales, es decir, todos los atributos y sin transformaciones para que sean comprensibles cada uno en su escala.

El primer dato que estudiaremos será el tamaño de cada grupo. La Figura 4.29 muestra gráficamente la cardinalidad de cada grupo.

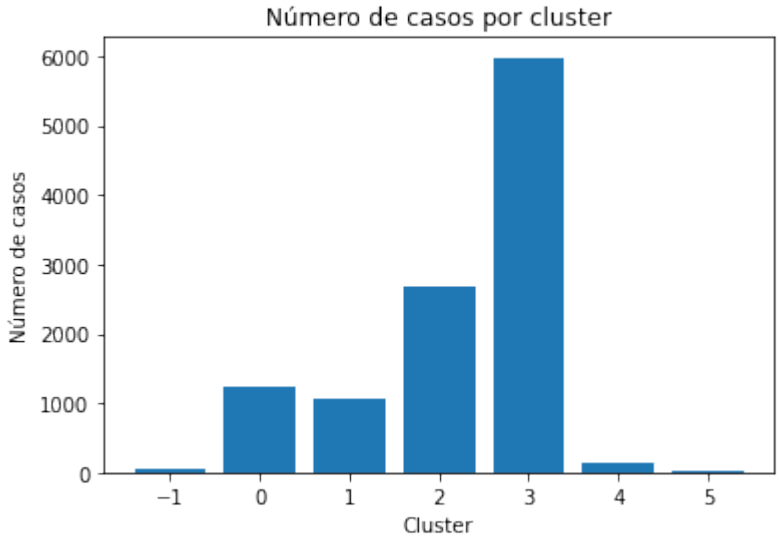

Figura 4.29: Número de casos por cluster

Analizando el gráfico, diferenciamos que el grupo 3 es el más grande, mientras que el 5 es el más pequeño. Los grupos 4 y 5 son prácticamente inexistentes, esto queda explicado en la poca diferencia del índice Silhouette a la hora de hacer 4 o 6 grupos. Además, al tratarse del algoritmo DBSCAN, tenemos el grupo de los puntos etiquetados como ruido. Estos no se van a tener en cuenta para el estudio de los grupos (ver Figura 4.30).

A continuación, se han calculado la media como valor estadístico descriptivo de cada variable de cada grupo.

	0	1	2	3	4	5
age	14.057143	14.116257	14.29013	14.089829	13.590278	13.0
total_ok	461.049796	325.481096	407.297207	213.680328	59.361111	55.8
total_error	491.319184	241.217391	361.932216	99.467882	0.0	0.0
total_tip	10.977959	0.0	10.788454	0.0	0.0	2.3
total_wiki	16.702857	8.225898	0.0	0.0	0.0	0.0
total_time	67969.805714	29782.801512	63966.221229	20843.854634	1960.520833	1967.2
acc	0.565464	0.637501	0.622128	0.763271	1.0	1.0

Figura 4.30: Valores de las medias de cada variable por grupos

.

Analizando los valores de la tabla, se han sacado las siguientes conclusiones:

- La edad, en términos medios, no es una factor diferencial, ya que en todos los grupos hay una media de edad parecida.

- El alumnado que más tiempo han pasado en la plataforma son los que más tareas han hecho correctamente, sin embargo, no son los que mejor desempeño tienen. Por ejemplo, el alumnado del grupo 0 son los más implicados, los que más tiempo han pasado y los que mayor número de tareas correctas tienen realizadas. Sin embargo, son los que más errores han cometido, los que más pistas han solicitado y los que más han consultado los apuntes. Esto queda reflejado precisión (`acc`) de este grupo, que es el más bajo.

- El grupo 1 tiene un desempeño que llama la atención, teniendo el cuenta el objetivo de este proyecto. Tienen la segunda mejor precisión y no han solicitado ninguna pista. Sin embargo, han accedido a los apuntes más que la media. Esto puede ser una señal de dificultad en la retención de la información o memoria.

- Los datos de los grupos 4 y 5 son los vamos a tener en cuenta por su pequeña cardinalidad.

CAPÍTULO 5

Seguimiento y control

En este apartado se han medido las desviaciones entre el tiempo estimado y el tiempo real dedicado al proyecto, así como la desviación en los hitos marcados.

5.1. Desviaciones

Ante un proyecto de esta magnitud, las desviaciones en la estimación de recursos es inevitable. Al inicio de este proyecto, se hizo una estimación y planificación para ejecutar el proyecto de la mejor forma posible atendiendo a las exigencias que tiene un Trabajo de Fin de Grado. Esta anticipación permite, además de organizar el trabajo, calcular la desviación y analizar los puntos críticos una vez acabado el proyecto con el objetivo de aprender y adaptarnos para planificar mejor los futuros proyectos.

La comparación y el cálculo de la desviación de las horas esta representada en la Tabla 5.1.

Código	Tarea	Estimación	Real	Desviación	
Gestión					
G.1	Plafinicación	10 h	12 h	+2 h	+20 %
G.2	Seguimiento	15 h	15 h	+5 h	+0 %
G.3	Control	5 h	7 h	+2 h	+40 %
		30 h	**34 h**	**+4 h**	**+13 %**
Investigación					
I.1	Contexto	15 h	22 h	+7 h	+47 %
I.2	Estudio de las herramientas	25 h	20 h	-5 h	+20 %
		40 h	**42 h**	**+2 h**	**+5 %**
Desarrollo					
D.1	Recolección y estudio de los datos	15 h	15 h	+0 h	+0 %
D.2	Preprocesamiento de los datos	30 h	15 h	-15 h	-50 %
D.3	Transformación, escalado y normalización	20 h	15 h	-5 h	-25 %
D.4	PCA	10 h	7 h	-3 h	-33 %
D.5	Clustering	15 h	23 h	+8 h	+53 %
D.6	Análisis de resultados	15 h	10 h	-5 h	-33 %
		100 h	**85 h**	**-15 h**	**-15 %**
Documentación					
DOC.1	Memoria	100 h	135 h	+35 h	+35 %
DOC.2	Anexos y referencias	5 h	9 h	+4 h	+80 %
DOC.3	Presentación	20 h	25 h	+5 h	+25 %
		125 h	**169 h**	**+44 h**	**+35 %**
Total		**295 h**	**330 h**	**+30 h**	**+30 %**

Tabla 5.1: Cálculo de las desviaciones por paquete de trabajo y tarea.

En general, las desviaciones son aceptables para los paquetes de trabajo de Gestión, Investigación y Desarrollo. Sin embargo, hemos necesitado más tiempo de lo previsto para la Documentación. Esto ha sido debido a, por un lado, la búsqueda de referencias científicas para cada concepto teórico mencionado en la memoria y por otro lado, intentar cuidar la coherencia, estructura y formato de la memoria. Ver la sección 5.4 para ver todos los problemas encontrados durante el desarrollo del proyecto.

5.2. Diagrama Gantt real

Los tiempo de finalización de cada tarea y paquete de trabajo ha cambiado respecto a lo previsto y el diagrama final de Gantt tiene esta forma (ver figura 5.1).

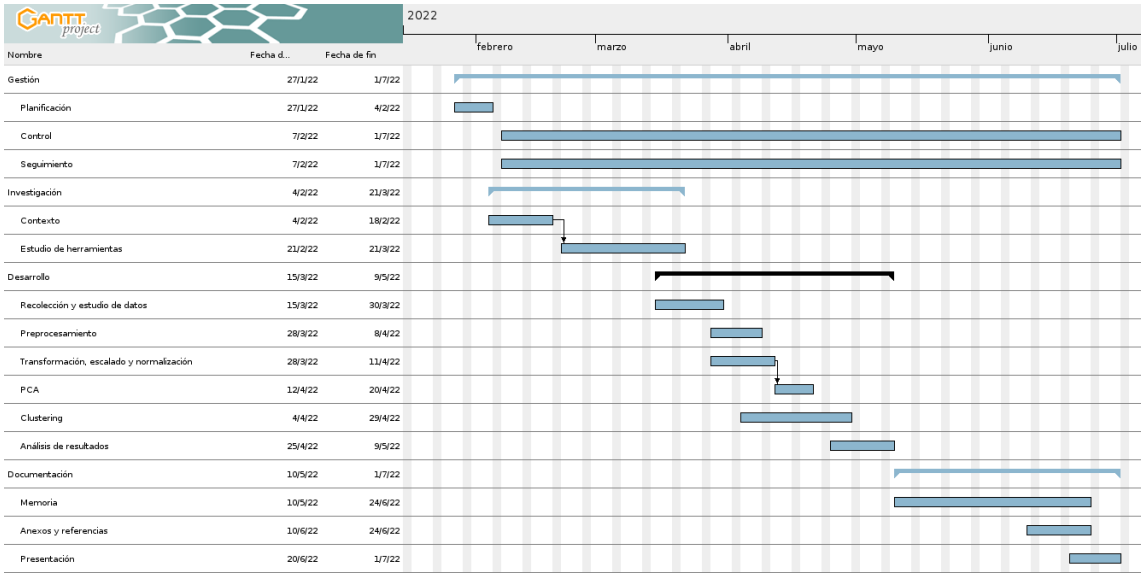

Figura 5.1: Diagrama Gantt con fechas de finalización post desarrollo

5.3. Hitos reales

En esta sección se comparan las fechas de los hitos previstos con las fechas en las que han finalizados esos hitos después de concluir el proyecto (ver tabla 5.2).

Objetivo	Fecha estimada	Fecha real
Desarrollar planificación	25-01-2022	04-02-2022
Base de conocimiento	25-02-2022	21-03-2022
Dataset	23-03-2022	11-04-2022
Implementación de los algoritmos	01-04-2022	29-04-2022
Redacción de la memoria	25-05-2022	24-06-2022
Presentación para la defensa	24-06-2022	En proceso

Tabla 5.2: Comparación de la fecha de los hitos iniciales con las fechas de finalización reales.

A la hora de la entrega de la memoria, la presentación para la defensa esta en proceso, es por ello que hemos podido especificar la fecha de finalización de este hito.

5.4. Problemas encontrados

Como se ha mencionado en la sección 5.1, hemos tenido más retraso con la Documentación. Esto ha sido principalmente por dos fractores:

- La estructura, coherencia y formato de la memoria ha sido objetivo que hemos querido mantener estrictamente y esto ha llevado a dedicar más tiempo de los esperado a revisar todos los detalles. Entre los detalles se encuentran intentar referencias en el texto todas la figuras y tablas o encontrar la referencias bibliográficas adecuadas para concepto teórico.

- Creación de una documentación adecuada del código en el `notebook` que reúne toda la programación utilizada en este proyecto. Aunque en la memoria se describe cada proceso y los resultados, aprovechando la estructura de los *notebook*, se han añadido explicaciones adicionales de cuestiones más especificas del código, además de los comentarios que permite Python.

CAPÍTULO 6

Conclusiones

Para finalizar el proyecto, he sacado varias conclusiones sobre el trabajo realizado y las herramientas utilizadas. Además, he aprendido varias lecciones que considero valiosas para mi futuro profesional y académico. Junto a ello, he definido algunas mejoras y futuras lineas de trabajo.

6.1. Aprendizajes

Al ser la primera vez que he llevado a cabo un proyecto de esta magnitud, he tenido la oportunidad de aprender a la vez que he desarrollado el proyecto.

Primero, la correcta autogestión del trabajo y del tiempo creo me ha aportado una experiencia muy enriquecedora para los futuro proyectos en los que vaya a participar. El hábito de organización y medición del tiempo para su posterior análisis creo que es un aprendizaje que lo aplicaré con frecuencia en el futuro.

Además, la búsqueda de información necesaria y adecuada creo es una competencia que he desarrollado mucho durante este proyecto. Me he familiarizado con la bibliografía científica y he sido capaz, de una forma autodidacta, interiorizar y estudiar conceptos utilizando estos recursos digitales (artículos, vídeos, transparencias, entre otras). No cabe duda de que he asentado una base de conocimiento sólida sobre el procesamiento y transformación de datos, y la implementación y aplicación de algoritmos de agrupación.

Por otro lado, el lenguaje de programación Python me ha permitido implementar y utilizar todas las técnicas necesarias para hacer un análisis adecuado de los datos.

6.2. Mejoras y futuras líneas de trabajo

En primer lugar, considero una mejora a corto plazo en la plataforma Toolbox.Academy la utilización del conocimiento obtenido en este análisis para enriquecer los informes que los profesores pueden generar automáticamente desde la plataforma. En estos informes se presenta la calificación que ha obtenido cada estudiante en cada módulo realizado. Además de esta información, creo que añadir como calificación cualitativa el grupo al que pertenecería el usuario teniendo en cuenta su rendimiento y utilizando este modelo, podría ayudar al profesor a crear un perfil de estudiante más completo.

Por otro lado, definiría como futura línea de trabajo la implementación de un *Intelligent Tutoring System* para que el recorrido que haga cada usuario a través de los contenidos de la plataforma se adapte a su nivel, pueda progresar adecuadamente y reciba una atención más personalizada.

Anexos

ANEXO A

Código del proyecto

El código que ha sido utilizado en el proyecto esta disponible en https://github.com/angalbi/clustering_toolbox.git. El formato que se ha trabajado durante todo el proyecto ha sido el *notebook*, con lo cual, todo el código esta en este formato con su correspondiente documentación y explicaciones.

Bibliografía

[Abdi and Williams, 2010] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459.

[Dašić et al., 2016] Dašić, P., Dašić, J., Crvenković, B., and Šerifi, V. (2016). A review of intelligent tutoring systems in e-learning. *Annals of the University of Oradea*, 3:85–89.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). 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.

[Google, 2021] Google (2021). Clustering in machine learning: Prepare data. https://developers.google.com/machine-learning/clustering/prepare-data.

[Johnson, 1967] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

[Ketchen and Shook, 1996] Ketchen, D. J. and Shook, C. L. (1996). The application of cluster analysis in strategic management research: An analysis and critique. *Strategic Management Journal*, 17(6):441–458.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations.

[Moreno Sandoval, 2018] Moreno Sandoval, L. G. (2018). Data mining techniques applied in educational environments: Literature review. *Digital Education Review*, 33:2018.

[Phobun and Vicheanpanya, 2010] Phobun, P. and Vicheanpanya, J. (2010). Adaptive intelligent tutoring systems for e-learning systems. *Procedia - Social and Behavioral Sciences*, 2(2):4064–4069. Innovation and Creativity in Education.

[Regueras et al., 2019] Regueras, L. M., Verdú, M. J., De Castro, J.-P., and Verdú, E. (2019). Clustering analysis for automatic certification of lms strategies in a university virtual campus. *IEEE Access*, 7:137680–137690.

[Romero and Ventura, 2007] Romero, C. and Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135–146.

[Rousseeuw, 1987] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

[SaS, 2022] SaS (2022). Etl. what it is and why it matters. https://www.sas.com/en_us/insights/data-management/what-is-etl.html.

[Vico, 2020] Vico, F. (2020). Toyscript: A computer language for teaching coding in the k-12 system.

[Vico et al., 2019] Vico, F., Masa, J., and García, R. (2019). Toolbox.academy: Coding & artificial intelligence made easy for kids, big data for educators. In *EDULEARN19 Proceedings*, 11th International Conference on Education and New Learning Technologies, pages 5173–5178. IATED.

[Xu and Tian, 2015] Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.