



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Herramienta docente para la
visualización en Web de
algoritmos de aprendizaje
Semi-Supervisado
Documentación técnica**



Presentado por David Martínez Acha
en Universidad de Burgos
el 11 de junio de 2023

Tutores: Dr. Álvar Arnaiz González
Dr. César Ignacio García Osorio

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	35
Apéndice B Especificación de Requisitos	43
B.1. Introducción	43
B.2. Objetivos generales	43
B.3. Usuarios de la aplicación	44
B.4. Catálogo de requisitos	44
B.5. Especificación de requisitos	47
Apéndice C Especificación de diseño	67
C.1. Introducción	67
C.2. Diseño de datos	67
C.3. Diseño procedimental	76
C.4. Diseño arquitectónico	78
C.5. Diseño Web	82
Apéndice D Documentación técnica de programación	87
D.1. Introducción	87

D.2. Estructura de directorios	87
D.3. Manual del programador	91
D.4. Compilación, instalación y ejecución del proyecto	99
D.5. Pruebas del sistema	103
Apéndice E Documentación de usuario	123
E.1. Introducción	123
E.2. Requisitos de usuarios	123
E.3. Instalación	123
E.4. Manual del usuario	123
E.5. Manual del administrador	146
Bibliografía	151

Índice de figuras

A.1. Burndown chart del sprint 8.	9
A.2. Burndown chart del sprint 9.	11
A.3. Burndown chart del sprint 10.	13
A.4. Burndown chart del sprint 11.	16
A.5. Cálculo del gasto de la empresa por empleado	36
A.6. Cálculo de la amortización para este proyecto	37
A.7. Permisividad de las licencias.	40
 B.1. Diagrama de casos de uso	 48
C.1. Diagrama Entidad/Relación	75
C.2. Diagrama relacional	76
C.3. Diagrama de secuencia de la visualización de un algoritmo . . .	77
C.4. Arquitectura de tres capas.	78
C.5. Arquitectura de tres capas.	80
C.6. Diagrama de componentes.	81
C.7. Diagrama de despliegue.	81
C.8. Página inicial de la Web.	82
C.9. Página de configuración del algoritmo.	83
C.10. Página de ejecución del algoritmo.	84
C.11. Barra de navegación principal	84
C.12. Barra de navegación con usuario	84
C.13. Animación «hover» en barra de navegación	84
C.14. Espacio personal del usuario	85
C.15. visualización del perfil y modificación	85
 D.1. Entorno Selenium IDE	 104
 E.1. Flujo de la visualización de un algoritmo	 124

E.2. Página principal	125
E.3. Carga del conjunto de datos	126
E.4. Condiciones de entrada	127
E.5. Configuración del algoritmo	128
E.6. Vista general de una visualización	129
E.7. Visualización principal	130
E.8. Tooltip con dato inicial	131
E.9. Tooltip con datos solapados	132
E.10.Tooltip con datos solapados con uno inicial	132
E.11.Tooltip con un dato no clasificado	133
E.12.Tooltip con un dato clasificado	133
E.13.Tooltip con un dato no clasificado	133
E.14.Tooltip con un dato clasificado	134
E.15.Tooltip con datos solapados y clasificados	134
E.16.Tooltip con un dato clasificado por dos clasificadores	135
E.17.Tooltip con datos solapados y clasificados	136
E.18.Gráficos estadísticos	137
E.19.Desplegable pseudocódigo	138
E.20.Cambiar de idioma	139
E.21.Formulario de registro	140
E.22.Formulario de inicio de sesión	140
E.23.Cierre de sesión	141
E.24.Acceso al perfil	141
E.25.Perfil personal y edición	142
E.26.Acceso al espacio personal	142
E.27.Espacio personal	143
E.28.Selección de algoritmo	144
E.29.Eliminar fichero	144
E.30.Parámetros de una ejecución	145
E.31.Eliminar ejecución	146
E.32.Acceso al panel de administración	147
E.33.Administración de usuarios	147
E.34.Conjuntos de datos subidos	148
E.35.Historial de ejecuciones	148
E.36.Edición de un perfil ajeno	149
E.37.Eliminación de usuario	150

Índice de tablas

A.1. Tabla de impuestos	36
A.2. Salarios brutos y coste que supone a la empresa	36
A.3. Costes del hardware	37
A.4. Costes indirectos	38
A.5. Coste total	38
A.6. Bibliotecas usadas, versiones y licencias	39
B.1. Caso de uso 1: Visualizar un algoritmo.	49
B.2. Caso de uso 2: Seleccionar algoritmo.	50
B.3. Caso de uso 3: Cargar conjunto de datos.	51
B.4. Caso de uso 4: Configurar algoritmo.	52
B.5. Caso de uso 5: Controlar visualizaciones.	53
B.6. Caso de uso 6: Registrar.	54
B.7. Caso de uso 7: Iniciar de sesión.	55
B.8. Caso de uso 8: Iniciar de sesión.	56
B.9. Caso de uso 9: Personalizar perfil.	57
B.10.Caso de uso 10: Acceder espacio personal.	58
B.11.Caso de uso 11: Ejecutar conjunto de datos.	59
B.12.Caso de uso 12: Replicar ejecución.	60
B.13.Caso de uso 13: Acceder panel de administración.	61
B.14.Caso de uso 14: Eliminar conjunto de datos.	62
B.15.Caso de uso 15: Eliminar ejecución.	63
B.16.Caso de uso 16: Editar perfil ajeno.	64
B.17.Caso de uso 17: Eliminar usuario.	65
B.18.Caso de uso 18: Cambiar idioma.	66
B.19.Caso de uso 19: Consultar ayuda.	66
C.1. Ejemplo de DataFrame de Self-Training	69

C.2. Ejemplo de DataFrame de Co-Training	70
C.3. Ejemplo de DataFrame de Democratic Co-Learning	71
C.4. Ejemplo de DataFrame de Tri-Training	72
C.5. Ejemplo de DataFrame de las estadísticas generales	73
D.1. Caso de prueba 1: Registro.	105
D.2. Caso de prueba 2: Inicio de sesión.	106
D.3. Caso de prueba 3: Cambio de idioma.	107
D.4. Caso de prueba 4: Edición perfil.	108
D.5. Caso de prueba 5: Edición perfil con contraseña incorrecta.	109
D.6. Caso de prueba 6: Usuario sin acceso a administración.	110
D.7. Caso de prueba 7: Anónimo sin acceso a administración.	110
D.8. Caso de prueba 8: Fichero sin extensión ARFF o CSV.	111
D.9. Caso de prueba 9: Ejecución correcta Self-Training.	112
D.10.Caso de prueba 10: Ejecución correcta Co-Training.	114
D.11.Caso de prueba 11: Ejecución correcta Democratic Co-Learning.	116
D.12.Caso de prueba 12: Ejecución correcta Tri-Training.	117
D.13.Caso de prueba 13: Error con atributos categóricos.	118
D.14.Caso de prueba 14: Borrar conjunto de datos.	119
D.15.Caso de prueba 15: Borrar ejecución.	120
D.16.Caso de prueba 16: Edición perfil (administrador).	121
D.17.Caso de prueba 17: Borrar perfil (administrador).	122

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En el presente apartado de los anexos se analizará la gestión del proyecto software desarrollado. Este proyecto será organizado mediante la metodología Scrum en la que el trabajo estará dividido en Sprints. Por cada Sprint se realiza una reunión para la revisión del avance y los objetivos para el siguiente. Con esta metodología se mantendrá en todo momento lo que se conoce como *Product Backlog* que es una lista de las tareas a realizar. Esta lista será actualizada en cada reunión para mantener un desarrollo constante.

Las reuniones, en un principio, se realizan cada dos semanas, intensificando a cada semana en el momento del inicio del periodo temporal del segundo cuatrimestre.

Finalmente también se incluirá un análisis de la viabilidad económica y legal (ámbito de licencias software).

El objetivo de este plan es servir como herramienta para registrar el avance del proyecto y también para poder cumplir con el objetivo final del desarrollo.

A.2. Planificación temporal

La planificación temporal se comenzó mediante Sprints de dos semanas. En la presente sección se comentará el desarrollo realizado en cada uno de ellos.

Sprint 0

Desde el punto de vista temporal, corresponde desde el inicio del curso del primer cuatrimestre académico (septiembre) hasta el Sprint 1. El día 15 de septiembre se tuvo la primera reunión con los tutores sobre el trabajo presente donde se establecieron las líneas generales y temática sobre el mismo.

Se creó el repositorio del TFG en Github: <https://github.com/dma1004/TFG-SemiSupervisado> y se añadió la plantilla de la documentación.

Sprint 1

Corresponde con el periodo temporal del 5 al 19 de octubre de 2022.

El mismo día 5 tuvo lugar una reunión de seguimiento del trabajo. Durante el sprint se realizaron unos arreglos de la plantilla y una lectura de conceptos teóricos para posteriormente añadirlos a la documentación. Concretamente se crearon las tareas «Añadir conceptos teóricos aprendizaje» y «Trabajos relacionados» a día 9 de octubre.

Sprint 2

Corresponde con el periodo temporal del 19 de octubre al 2 de noviembre de 2022.

Durante el sprint se implementó un prototipo del algoritmo Self-Training en el que posteriormente se hicieron unas correcciones en el código. También se comenzó con la redacción de conceptos teóricos, concretamente, sobre el aprendizaje automático.

Sprint 3

Corresponde con el periodo temporal del 16 al 30 de noviembre de 2022.

Durante el sprint se aumentaron los conceptos teóricos sobre el aprendizaje supervisado, no supervisado y semi-supervisado. Se refactorizó el prototipo para su documentación (PEP), evitar datos duplicados y modularizar el código.

La memoria fue parcialmente modificada basándose en las correcciones propuestas de los tutores.

Sprint 4

Corresponde con el periodo temporal del 25 de enero al 1 de febrero de 2023. En este momento las duraciones de los Sprints cambiaron a una semana, iniciando así el periodo temporal real del desarrollo del proyecto (segundo cuatrimestre).

Durante el sprint se retomaron las tareas y el desarrollo general del proyecto. Se mejoró el algoritmo de *Self-Training* que estaba como prototipo y se avanzó en la tarea de la primera aproximación en la aplicación mediante el *framework* Flask. Sobre esto último, se creó una visualización del proceso de entrenamiento muy básica por cada iteración.

Se creó un prototipo del algoritmo *Co-Training* sin cumplir con todas sus condiciones que posteriormente se completaron a falta de revisión.

Sobre estos dos algoritmos se propuso la versión 1.0.

Continuando con la web, se realizó la interfaz general funcional. Incluye:

- Página de Inicio donde seleccionar el algoritmo.
- Página de subida de archivos en formatos «ARFF» y «CSV» de los conjuntos de datos
- Páginas correspondientes para *Self-Training* y *Co-Training*: Cada una tiene sus parámetros específicos con la posibilidad de seleccionar si utilizar PCA (*Principal Component Analysis*) o dos componentes que elija el usuario.
- Página de visualización del algoritmo (su entrenamiento): Se tiene la vista principal que será común a todos los algoritmos (con algunas variaciones en caso necesario), con la posibilidad de avanzar en la visualización (con controles) y barra de progreso. Desde el punto de vista del gráfico los colores están automatizados dependiendo del número de clases, leyenda y etiqueta de ejes.

En Flask se añadieron los «endpoints» correspondientes (subida, configuración, visualización...) y un control de acceso a las páginas muy básico (por ejemplo, si no se configuró el algoritmo, no se puede visualizar y le redirecciona a la configuración con un mensaje de error)

Sprint 5

Corresponde con el periodo temporal del 1 al 8 de febrero de 2023.

En la reunión del 1 de febrero se revisó lo realizado en el anterior y se fijaron una serie de mejoras/modificaciones y nuevas tareas:

1. Modificación de los algoritmos para trabajar con la convención de utilizar la etiqueta «-ls» en el conjunto de datos para los datos no etiquetados. Así el usuario podrá subir un archivo que tenga datos etiquetados y no etiquetados.
2. Permitir al usuario seleccionar los porcentajes de no etiquetados y de test (para las futuras estadísticas).
3. Sobre la página general de la visualización de los algoritmos: volver a la configuración, el «feedback» de la iteración actual y el nombre del conjunto de datos utilizado.
4. Del gráfico de la visualización: Diferenciar en el algoritmo *Co-Training* cuál de los dos clasificadores han etiquetado cada punto y los puntos «etiquetados» en la iteración 0 deben mostrarse de forma diferente.
5. Avanzar con los trabajos relacionados.
6. Avanzar con la documentación teórica y anexos.

El punto 1 ha llevado unas 12 horas de compresión y desarrollo. Esto es debido a que los dos algoritmos implementados hasta ahora debían ser modificados para trabajar con la nueva convención. Además, el problema principal fue (aunque no implementado en este Sprint) dejar preparada una forma de carga del conjunto de datos que permita tratar datos no etiquetados (los «?» en el caso de ARFF), pues además de los algoritmos (su correcto funcionamiento), se han probado con ficheros. También conllevó la creación de un codificador de etiquetas propio para ignorar los no etiquetados en clases categóricas (y no realizar la conversión en esos casos).

El punto 2 volvió a causar bastantes problemas tanto en la ejecución de los algoritmos como en la web. Hasta el momento, el usuario no seleccionaba los porcentajes de las divisiones, pero al incluir esto, los algoritmos ya no se encargan de esta tarea y había que modificar tanto los algoritmos como aquellas rutas de la web que debían encargarse de esto. Aproximadamente 4 horas.

El punto 3 no resultó demasiado difícil más allá de seguir habituándose a JavaScript/HTML. Unas 3 horas.

El punto 4 requirió unas 10 horas, en un principio se perdió mucho tiempo intentando solucionarlo de una forma que resultó inútil, pero finalmente ahora en el algoritmo se diferencian los datos clasificados por cada uno.

Los trabajos relacionados (no terminados) se realizaron en varios días con un tiempo aproximado de 6 horas.

Sprint 6

Corresponde con el periodo temporal del 8 al 15 de febrero de 2023.

En la reunión del 8 de febrero se revisó lo realizado en el anterior y se comentaron algunas tareas a realizar:

1. En la línea del anterior, los algoritmos deben poder ejecutarse directamente con conjuntos de datos semi-supervisados.
2. Permitir al usuario introducir ese tipo de conjuntos de datos.
3. Realizar alguna visualización de estadísticas.
4. Valor por defecto en las configuraciones.
5. Sobre el gráfico: mejorar la diferenciación de los puntos, información útil en los «*tooltips*» y colocación de la leyenda.
6. Avanzar con los trabajos relacionados.
7. Avanzar con la memoria y anexos.

Los puntos 1 y 2 estaban muy avanzados gracias al trabajo adicional del sprint anterior, ya que estaba prácticamente implementada la forma en la que detectar datos no etiquetados de forma automática. Unas 5 horas para terminar de implementar, corregir errores sobre la marcha y realizar alguna prueba confeccionando ficheros semi-supervisados.

Al realizar las pruebas anteriores se encontró un error en la visualización provocando que los datos que los datos no se habían clasificado ni siquiera eran retornados a la Web. Entre descubrir cómo corregirlo y sus modificaciones se tardó unas 3 horas.

El punto 3 fue el más complicado, pese a que era una idea sencilla, se optó por visualizar la gráfica de la evolución de la precisión. Cada punto del

gráfico está unido por una serie de líneas. Este tipo de gráficos (según la documentación) se suelen hacer mediante «*paths*» o caminos, que son una única línea, pero como en este caso era necesario no visualizar todo, sino por cada iteración, no se encontró una solución rápida. Unas 6 horas para probar muchas posibilidades hasta encontrar la que funcionó, acoplarla a los controles del paso de iteración e incluir alguna animación.

Adicionalmente se retocó por completo toda la Web mediante los estilos de **Bootstrap** para establecer ya una base vistosa y bonita. Unas 5 horas (la mayor parte del tiempo para probar y adquirir algo de soltura con estos estilos).

Sprint 7

Corresponde con el periodo temporal del 15 al 22 de febrero de 2023.

Puntos a desarrollar:

1. Implementación *Democratic Co-Learning*.
2. Profiling (tiempos de ejecución).
3. Estadísticas en la aplicación.
4. Test de las implementaciones.
5. Avanzar con la memoria y anexos.

La implementación del algoritmo *Democratic Co-Learning* supuso unas 14 horas divididas en varios días. Al principio se dedicó un tiempo para leer el artículo en el que se presentaba su implementación en forma de pseudocódigo junto con sus explicaciones teóricas. La realidad es que en primera instancia parecía algo fácil de realizar y entender, pero una vez comenzada la implementación, se encontraron muchas complicaciones a la hora de hacerlo.

Además, pese a que en el artículo estaba bien explicado, el formato de pseudocódigo (en el archivo encontrado) tenía indentaciones incorrectas y se perdió mucho tiempo comprobando si era una interpretación errónea o si realmente era un fallo.

Se realizaron algunas pruebas de rendimiento para comprobar si los algoritmos tardaban demasiado con conjuntos de datos muy grandes (5 000 instancias). Se observó que, dada la configuración que se tenía, tardaba

alrededor de 40-50 segundos en terminar la ejecución. Es por esto que para este sprint se añadió la tarea de hacer un pequeño estudio dedicado a medir los tiempos de ejecución para ver qué se podía optimizar.

Este proceso fue de unas 2 horas y el resultado fue que el código implementado no afectaba mucho, eran los propios algoritmos de entrenamiento de los clasificadores de *Scikit-Learn* los que tardaban tanto. Por ejemplo, para un estimador gaussiano el tiempo se reducía drásticamente.

Para el caso de las estadísticas, se modificaron un poco las plantillas y la generación de sus gráficas para incluir más y revisarlas en la reunión. Unas 2 horas.

Los tests son una parte importante para validar que el comportamiento que se espera de la implementación sea el correcto. Se realizaron unos casos de pruebas sobre las utilidades que se usan a lo largo de todo el proyecto con la intención de encontrar errores (todo esto sin ver cuál es el resultado y replicarlo en los casos, sino realizar los casos basándose en lo que se espera de esas utilidades). Se tardó unas 4 horas en realizar todos los tests.

Sprint 8

Corresponde con el periodo temporal del 22 de febrero al 1 de marzo de 2023. Además, aprovechando la herramienta Zenhub, se modificó la duración de los Sprints también en ella para poder extraer los gráficos del trabajo realizado.

Puntos a desarrollar:

1. Intervalo de confianza en *Democratic Co-Learning*.
2. Control reetiquetado en *Democratic Co-Learning*.
3. Correcciones sobre memoria y anexos.
4. Gráfico de estadísticas unificado.
5. Internacionalización web.
6. Visualización principal de *Democratic Co-Learning* en la aplicación.

El primer punto fue muy sencillo, se proporcionó la implementación de los intervalos de confianza tanto de Álar Arnaiz González como de César Ignacio García Osorio (tutores) y finalmente se implementó esta segunda.

El control del reetiquetado fue mal estimado (en puntos de historia), al principio parecía una idea sencilla, pero por un error de pensamiento, la implementación que se realizó en un principio no funcionaba. Como cada clasificador tiene su propio conjunto de entrenamiento, se estaba tomando que el índice de la instancia sumado a la longitud de su conjunto de entrenamiento era la posición en la que actualizar la etiqueta, obviamente esto no funciona pues esa suma puede superar la longitud del propio conjunto. Había que almacenar la posición concreta sin realizar esos cálculos. Se optó por un diccionario de identificadores para cada clasificador en el que el valor es la posición en las etiquetas del conjunto del clasificador.

Las correcciones de la documentación se incorporaron según las indicaciones de Álar Arnaiz.

El gráfico de estadísticas fue unificado, permitiendo seleccionar al usuario mediante unos «checkboxes». Aproximadamente unas 4 horas para generar el formulario correspondiente que controle la aparición de cada línea y controlar los eventos de las iteraciones (por ejemplo, añadir siguiente punto solo si está activado su *check*).

En cuanto a la Internacionalización, al principio resultó sencillo, ya que Babel (Flask-Babel) detecta automáticamente las cadenas de texto dentro de `gettext`. Sin embargo, al indicar las traducciones en JavaScript, el intérprete tomaba `gettext` como una función que al no estar definida, lanzaba error. Al final se generó una función en la plantilla principal de tal forma que actúe como `gettext`, llamada desde los distintos scripts.

La visualización de *Democratic Co-Learning* no dio tiempo a implementarse en este sprint.

Además, a partir de este Sprint se incorporan todas las tareas en Zenhub para la generación de los gráficos *Burndown* (ver Imagen [A.1](#)).



Figura A.1: Burndown chart del sprint 8.

Sprint 9

Corresponde con el periodo temporal del 1 al 8 de marzo de 2023.

Puntos a desarrollar:

1. Visualización principal de *Democratic Co-Learning* en la aplicación.
2. Formulario de los parámetros de los clasificadores.
3. Estadísticas generales en *Democratic Co-Learning*.
4. Estadísticas específicas en *Democratic Co-Learning*.
5. Condición de parada re-etiquetado.

La visualización principal fue relativamente sencilla pues en realidad es muy similar a *Co-Training*. Hubo algún ajuste adicional para generar la información del *tooltip* al pasar por encima de un punto. Como en cada posición podía haber varios clasificadores había que mostrar esa información.

Para el formulario de los parámetros se consideró el uso de Flask-WTF que al final se descartó. Se tenía como punto de partida utilizar un JSON del que leer los parámetros, así que lo primero fue pensar una forma sencilla de codificarlos con la información necesaria de las entradas («*type*», «*step*», ...).

La ventaja de JSON es que aparte de que la lectura es muy sencilla, son diccionarios, muy fáciles de utilizar (tanto desde Python como desde las propias plantillas/JavaScript). Para generar el formulario se pensó en hacerlo de la forma más automática posible para así solo tener que cambiar el JSON en el futuro. Esto se hizo con un método de JavaScript que para cada clasificador genera el formulario correspondiente con los parámetros del JSON. El tiempo total fue unas 6 horas, pues también se perdió tiempo debido a que no era posible seleccionar elementos del DOM (pues no estaba cargado) y los elementos debían seleccionarse mediante CSS (algo que no se sabía por desconocimiento de JavaScript).

Se añadieron las estadísticas generales de *Democratic Co-Learning* de forma muy sencilla pues era exactamente igual que *Co-Training* (y *Self-Training*) esto se realizó añadiendo el *DataFrame* en el propio algoritmo con las estadísticas deseadas (como el resto de los algoritmos).

Había mucho código muy parecido o exactamente igual en las plantillas de los algoritmos, así que aprovechando la refactorización, se automatizaron por completo las estadísticas. Se pensó de tal forma que solo con las columnas de los *DataFrames* que retornan los algoritmos, la página web ya se encargue de generar el resto. A la vez que esto (e iniciando la tarea de las estadísticas individuales), se tuvieron en cuenta las futuras estadísticas individuales para *Democratic Co-Learning*. Todas las funciones fueron transformadas para trabajar con elementos del DOM específicos, de esta forma, al indicarle por ejemplo un «DIV», se generan las estadísticas sobre ese elemento. En total unas 8 horas.

Para estas estadísticas individuales, aparte de las funciones generalizadas, se creó una nueva que sobre un elemento (un «DIV») se generase un selector con el nombre de los clasificadores que intervienen en el algoritmo junto con los contenedores («DIV») dentro de él para las estadísticas de cada clasificador. Finalmente, se utilizan las funciones de la tarea anterior para añadir a esos contenedores nuevos los gráficos individuales (uno por cada clasificador). Unas 4 horas.

Sobre el último punto, se comentó que una etiqueta que es re-etiquetada al mismo valor no debe «contar» como mejora (o cambio en el algoritmo). Si no se realiza así, el tiempo de ejecución aumenta demasiado.

El gráfico Burndown se visualiza en la imagen [A.2](#).



Figura A.2: Burndown chart del sprint 9.

Sprint 10

Corresponde con el periodo temporal del 8 de febrero al 15 de marzo de 2023.

Puntos a desarrollar:

1. Comparación sslearn, una biblioteca de José Luis Garrido-Labrador [2].
2. Refactorización de plantillas.
3. Refactorización Javascript.
4. Refactorización Flask (app).
5. Añadir zoom a los gráficos.
6. Conceptos teóricos.

En este Sprint no se contaba con demasiado tiempo así que aunque sí se realizó alguna tarea para añadir funciones, la idea era dedicarlo a mejorar el código y mantenimiento.

En la reunión del final del sprint anterior se comentó el cómo se estaban validando los algoritmos y hasta ese momento solo se habían probado las

utilidades. Se sugirió compararlo contra *sslearn*. Como primera aproximación, se realizó una validación cruzada completamente manual en la que se ejecutaba al mismo tiempo las dos implementaciones de los distintos algoritmos (de momento sin extraer conclusiones). Se tardó unas 3 horas.

El segundo y tercer punto se iniciaron por separado, pero llegó un momento en el que las funciones que se había creado en JavaScript, si se modificaban un poco, podrían simplificarse las plantillas.

Todos los métodos de los gráficos estaban separados por cada uno de los algoritmos, esto lo hacía muy engorroso porque incluso algún método tenía el mismo nombre. Se modificaron las funciones de tal forma que fuesen específicas para cada algoritmo para así juntarlas en un único *script*. Por parte de las plantillas, como había partes repetidas se crearon macros y se identificó una parte común a todos los algoritmos en su configuración, esto se añadió a la base de las configuraciones. Todo esto llevó unas 4 horas.

En **Flask** se tenían varios problemas. El primero era que las visualizaciones de cada algoritmo tenían un *endpoint* particular, pero esto no era necesario si se hacía un método para la obtención de los parámetros de cada algoritmo por separado (y finalmente un solo *endpoint* indicándole el algoritmo en la propia ruta).

Cuando se crearon esos nuevos métodos se generó código muy parecido porque todos ellos tenían dos partes: una en la que se obtenían los parámetros que no eran de los clasificadores base y otra en la que se incorporaban esos parámetros de los clasificadores. La primera parte es particular para cada algoritmo, pero la segunda es común a todos. Se creó otro método para ese paso común.

Por último, por cada algoritmo se tiene un *endpoint* para la ejecución y obtención de la información de entrenamiento, pero había una parte que todos hacían prácticamente igual. Se creó un método que engloba: carga de datos, separación de los datos para entrenamiento, entrenamiento y obtención de los algoritmos y la aplicación de PCA (o no).

La visualización principal de los algoritmos tenía el problema de que cuando los puntos estaban demasiado cerca, no se llegan a apreciar individualmente. Esto se solucionaría aplicando *zoom* al gráfico. Pese a que en cuanto a código no fuese un desarrollo largo, fue una tarea compleja. Se probaron unas tres implementaciones parecidas a ejemplos encontrados en la documentación, pero en todas ellas se perdía la ayuda contextual del *tooltip*.

Al final se optó por volver a empezar de cero con el conocimiento adquirido y junto con un último ejemplo¹ y ciertas modificaciones, se consiguió.

Posteriormente se añadió el reinicio del *zoom* para volver a la posición original. El proceso duró unas 5 horas pues cada intento parecía ser definitivo, pero al final siempre había ciertos límites.

Al final del sprint se añadieron los conceptos teóricos de *Democratic Co-Learning* (conceptos y pseudocódigos).

Aparte de estas tareas se realizaron pequeñas modificaciones: se completó el estilo adaptable («responsive»), se arreglaron pequeños bugs de visualizaciones y ayuda contextual, actualización de traducciones y se añadió un fichero de prueba que el usuario puede descargar si solo quiere probar la aplicación.

El gráfico Burndown se visualiza en la imagen A.3.



Figura A.3: Burndown chart del sprint 10.

Sprint 11

Corresponde con el periodo temporal del 15 al 22 de marzo de 2023.

Puntos a desarrollar:

¹Ejemplo D3: <https://observablehq.com/@d3/zoom-with-tooltip>

1. Arreglos en *Democratic Co-Learning*.
2. Controlar los límites en los parámetros de los clasificadores.
3. Modificar las estadísticas generales.
4. Invertir controles en las estadísticas específicas.
5. Anexos: Manual del programador.
6. Modificar validación cruzada.
7. Comparación exhaustiva contra `sslearn`.
8. Añadir validación de los algoritmos a la memoria.

En *Democratic Co-Learning* se estaba obviando el caso en el que si una instancia ya estaba etiquetada y esta es reetiquetada, pero además cambiando de etiqueta, no se consideraba como cambio en el algoritmo. Se ha añadido esta casuística. Además, Álvar sugirió en el pseudocódigo de la memoria hacer el método de combinación de hipótesis (predicción) para una sola instancia. Esto se ha realizado así en la propia implementación (y así el método `predict` se encarga de iterar sobre un conjunto de instancias).

Al JSON que codifica los parámetros de los clasificadores base se añadieron los controles de mínimo y máximo. Esto es porque hay algunos de sus parámetros que requieren rangos específicos. Ahora la web (JavaScript) genera el formulario de configuración teniendo en cuenta estos límites.

En la reunión del sprint anterior se sugirió modificar la visualización de estadísticas. Para el gráfico general (de estadísticas) no tenía sentido poder ocultar o no cada una de las estadísticas así que ahora siempre se muestran todas ellas. Particularmente para *Democratic Co-Learning*, las estadísticas individuales estaban manejadas mediante un selector (para seleccionar el clasificador base) y unos `checkboxes` para seleccionar las estadísticas a mostrar. Pero tiene mucho más sentido que el selector sea para las estadísticas. De esta forma el usuario selecciona una estadística y en el gráfico puede comparar esa estadística para todos los clasificadores. Además, los `checkboxes` se mantienen, pero ahora sirven para elegir qué clasificadores comparar. Esto llevó unas 6 horas. La organización de las funciones estaba altamente centrada en la versión anterior, fue un proceso complicado y lioso.

Pese a que en un principio no se comentó, una vez que se terminó el punto anterior, se vio necesaria una reestructuración de la página de las

visualizaciones. Se mejoraron las leyendas de tal forma que ya no estaban dentro de los SVGs, ahora están en su propio cuadro. Esto ha conseguido no preocuparse por el tamaño de las palabras, centrarla y poder organizar mejor las columnas en las que se divide esa plantilla.

Sobre el manual del programador, se añadió la estructura de directorios con el paquete `dirtree` de \LaTeX y se completó el manual del programador, centrado en qué es lo que debe saber un desarrollador para continuar con el proyecto. Finalmente se describió el proceso de la compilación, instalación y ejecución del proyecto. En total fueron unas 6 horas.

El proceso de validación cruzada no estaba bien enfocado, se estaba realizando de forma manual (y todos los posibles fallos que puede suponer). La librería *Scikit-Learn* incorpora ya utilidades para este proceso. Concretamente se tiene un método que genera los distintos *folds* dado un conjunto de datos. El proceso manual se sustituyó por este nuevo. Se aprovechó para guardar los resultados como CSV.

Con el proceso de validación cruzada la idea era obtener métricas para compararlas en alguna gráfica/tabla y comprobar que las implementaciones son correctas (comparadas con `sslearn`). Para ello se creó una función que recogía la información de los CSVs y dibujaba una malla con distintos gráficos. En las columnas se distribuyen los algoritmos, separadas en dos para la implementación propia y la de `sslearn`. En las filas se tiene cada estadística. Cada uno de los gráficos de esta malla es un gráfico de cajas (para mostrar mínimos, máximos, medias, medianas...).

Sobre el último punto, aunque la idea era realizarlo en este Sprint, no se estimó bien la cantidad de trabajo y no pudo ser realizado. Además, en las ejecuciones del punto anterior (la comparativa) se vio que el algoritmo Co-Training tenía peor rendimiento que el de `sslearn`. Esto impedía justificar en la memoria la validación de los algoritmos.

Durante estas tareas fueron surgiendo pequeños arreglos: Se colorearon las etiquetas de la ayuda contextual (`tooltip`) para que quedara claro qué etiqueta lleva cada punto (no solo el nombre) y se actualizaron las traducciones.

El gráfico Burndown se visualiza en la imagen [A.4](#).

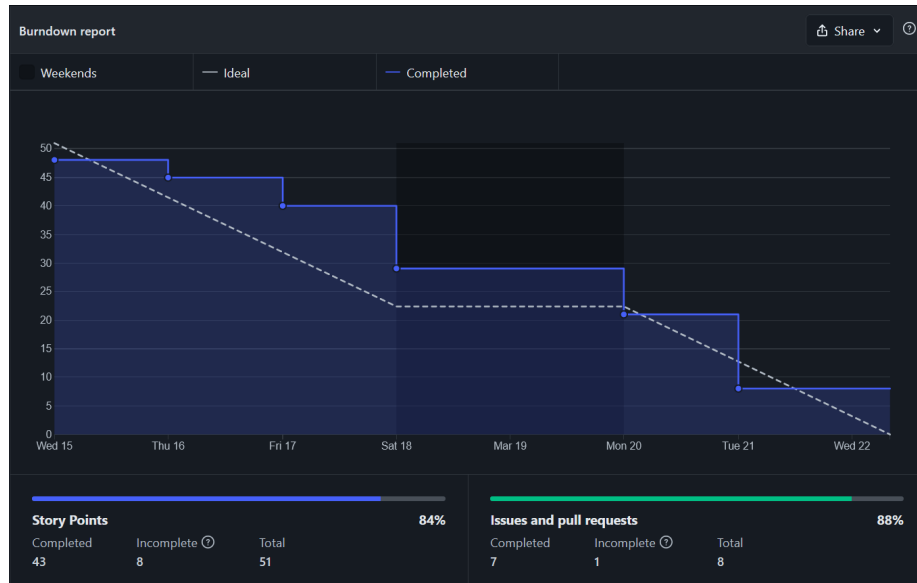


Figura A.4: Burndown chart del sprint 11.

Sprint 12

Corresponde con el periodo temporal del 22 al 29 de marzo de 2023.

Puntos a desarrollar:

1. Completar manual del programador.
2. Introducción memoria.
3. Añadir bibliotecas web.
4. Añadir explicaciones de los algoritmos en la web.
5. Añadir los pseudocódigos en la web.
6. Algoritmo *Tri-Training*.

En el sprint anterior se empezó el manual del programador, pero quedó pendiente la compilación, instalación y ejecución del proyecto. Lo primero que se hizo en este sprint es finalizar este manual describiendo todos los pasos que un programador debe seguir para poner en funcionamiento la aplicación.

Para continuar con la memoria, se añadió la sección de la introducción como presentación de todo este trabajo.

Las dos principales bibliotecas o recursos utilizados en la Web son **Bootstrap** y **D3.js**, el primer caso es un «*framework*» de CSS que ya viene con una gran cantidad de clases a utilizar. D3 es la biblioteca de JavaScript que se ha utilizado para la generación de todos los gráficos. Se añadió una descripción (y justificación) de uso en la memoria (Técnicas y herramientas).

Teniendo en cuenta la componente docente de este proyecto, se creyó conveniente la adición de unas pequeñas explicaciones de los algoritmos. Se añadieron en la fase de configuración de los algoritmos dado que al mismo tiempo el usuario debe configurar los parámetros y puede que le permita tener una mayor intuición a la hora de configurarlo.

En esta misma línea se añadieron los pseudocódigos en forma de imágenes, justo debajo de las explicaciones. También se creyó conveniente que el usuario pudiera verlo en la fase de visualización. Se creó un desplegable con la imagen del pseudocódigo.

Finalmente, y como desarrollo más grande de este sprint, se creó el algoritmo *Tri-Training*. La primera versión (el primer `commit`) no fue validada de ninguna forma (se confirmó según se terminó). Esta versión no incluía ninguna forma de obtener el proceso de entrenamiento, simplemente era el esqueleto funcional del algoritmo. Obviamente, era claro que iba a haber algún que otro error. Al principio solo se detectó un error en el que el cálculo del error de clasificación siempre resultaba en 0. Era una cuestión de pura implementación en Python así que no fue difícil arreglarlo.

Antes de arreglar este error, la rama en la que se ejecutaba el «*subsample*» no se estaba realizando en ningún caso. En el momento que se arregló el anterior error se accedió a esta parte y apareció un nuevo error. Se confundió array de NumPy con lista nativa de Python y se estaban seleccionando ciertas posiciones de una lista de Python accediendo a ella con unos índices (`List[indices]`). La solución fue sencilla de igual manera, se recorría la lista en base a los índices y se creaba esta sublista mediante compresión de listas.

Al final de este desarrollo, se comparó con una sola ejecución con `sslearn`. El nuevo algoritmo tardaba mucho más tiempo que el resto (y que `sslearn`). Esto era porque al añadir las predicciones (cuando las otras dos hipótesis coincidían) se hacía de una en una. Esto se sustituyó para predecir todas las instancias no etiquetadas y con vectorización. El rendimiento se incrementó mucho.

Durante estas tareas fueron surgiendo otras más pequeñas: Nuevas traducciones, correcciones de memoria y anexos y una pequeña comparación contra sslearn.

Licencia Zenhub

La herramienta Zenhub dejó de proporcionar servicios gratuitos. Esta compañía actualizó sus planes gratuitos, ya no existe una versión de esas características. Durante este Sprint «caducó» la licencia que se tenía y ya no se podía acceder a ninguna de sus funcionalidades, tampoco a los gráficos Burndown.

Sprint 13

Corresponde con el periodo temporal del 29 marzo al 5 de abril de 2023.

Puntos a desarrollar:

1. Configuración *Tri-Training*.
2. Visualizar *Tri-Training*.
3. Eliminar todo el código duplicado restante de JavaScript.
4. Documentar JavaScript.
5. Reestructurar Flask para aplicaciones grandes.
6. Añadir media geométrica en la validación/comparación contra sslearn.

Para este Sprint el objetivo principal era dejar lista la visualización de *Tri-Training* en la Web.

El primer paso fue incluir las estructuras de datos necesarias en el esqueleto del algoritmo para ir almacenando la información de entrenamiento. La idea era igual a *Democratic Co-Learning*. La realidad es que la forma en la que se almacenan los datos es muy distinta a todas las demás. Los datos pueden ser clasificados varias veces por cada clasificador en diferentes iteraciones. Por lo tanto, para cada clasificador base se almacena una lista para las etiquetas y para las iteraciones en las que se clasifican. Las estadísticas sí que se almacenan exactamente de la misma forma que *Democratic Co-Learning*.

El siguiente paso fue crear la pantalla de configuración de *Tri-Training*, para ello simplemente se copiaron las plantillas ya existentes y la misma estructura que ya se tenía en Flask para las rutas.

Y finalmente, la pantalla de visualización de *Tri-Training*. Este fue uno de los pasos más complejos. Como se tenía una nueva estructura de los datos, se tenían que crear los métodos que permitiesen interpretarlos, de forma general, los que se ha realizado es comprobar, en cada iteración, qué clasificador han añadido datos etiquetados nuevos junto con su etiqueta. Para que el usuario pudiera ver los cambios que ocurren en cada iteración, cuando pulsa en la siguiente iteración se descolorean los puntos (a gris) y después se colorean los nuevos, todo esto mediante animaciones y con un cierto tiempo para que pueda darse cuenta.

En cuanto a JavaScript, se documentaron por completo todos los métodos que fueron creados para este proyecto. Además, se eliminó el código duplicado en cada fichero utilizando métodos comunes. También fue un proceso largo, sobre todo para comprobar que las modificaciones eran correctas y generales, en principio, todo parece correcto.

Se reestructuró completamente Flask, esto es porque tal y como se encontraba la aplicación, era muy básica y general. Los proyectos más grandes con Flask tienen una estructura más o menos concreta. Esta es la idea que se ha intentado aplicar, utilizando «*Application Factory*» y «*Blueprints*». Pero sobre todo, para hacerla más mantenible y extensible. De hecho, aunque no está siendo usada, se tiene una pequeña base de datos para posibles adiciones futuras.

Esta reestructuración causó muchos problemas en cuanto a las rutas de ficheros. Se tuvieron que especificar manualmente y usar la librería del sistema operativo de Python para independizar ciertos tratamientos de ficheros del sistema operativo concreto donde se ejecute la aplicación (se ha probado en Windows y Linux).

Finalmente, se añadió la media geométrica como métrica de validación a la hora de comparar contra **sslearn** [2]. Además, en Sprints anteriores se comentó la idea de utilizar *violinplots*, que resultan más pertinentes para los casos de comparación.

Sprint 14

Corresponde con el periodo temporal del 5 al 12 de abril de 2023.

Puntos a desarrollar:

1. Actualizar manual del programador
2. Rediseño web completo

3. Ayuda contextual y errores en Web.

4. Anexos: Diseño

Debido a las modificaciones realizadas en los recientes sprints, el manual del programador quedó desactualizado. El cambio más notable fue la nueva reestructuración de Flask. Se actualizó el árbol de directorios reflejando esta nueva estructura. También se actualizaron todas las explicaciones posteriores del manual.

El diseño de la web estaba hecho prácticamente desde el principio de la web y resultaba muy poco llamativo. El primer paso para este rediseño fue elegir una paleta de colores, se creyeron convenientes fondos oscuros con alguno más claro para dar contraste.

Para continuar, se pensó en la disposición y estilo general del contenido. Hasta el momento la página resultaba muy plana, sin cambios entre las distintas partes de la página. Consultando páginas web de ejemplo parecía interesante y bonito crear efectos de sombra y se estableció como estilo general de la página. A partir de aquí las distintas secciones de una misma página se encuadran en un elemento con sombra, para llamar la atención del usuario. Además, para comprobar que no eran ideas descabelladas, se consultó a compañeros y personas externas para buscar opinión. Gracias a esto se realizaron retoques (cambio de algún color, títulos, disposiciones...) que no convencía a la mayoría.

Para proporcionar ayuda básica en las distintas páginas, se creó una macro que genera una `tooltip` con un mensaje. Tanto en la subida como en la configuración se utilizaron estos mensajes para aclarar aspectos confusos. Como adición, resultaba muy molesto tener que volver a subir un fichero constantemente al seleccionar otro algoritmo durante la misma sesión. Se añadió una comprobación en la pestaña de subida en la que si ya se había subido un fichero, permitiera ir directamente a la configuración.

Durante todo el desarrollo hasta este momento, ocurrían ciertos errores HTTP (404, por ejemplo) pero no eran controlados. Se creó una plantilla base y se le indicó a Flask los errores que debía manejar y renderizar en esa plantilla. Se han añadido los más comunes (y además muchos de ellos no han llegado a ocurrir nunca).

En reuniones anteriores se comentó que debía explicarse las estructuras de los ficheros JSON que maneja e interpreta la web (JavaScript). Se añadieron todas las explicaciones relativas a la generación de estos ficheros desde la propia ejecución de cada algoritmo. Se creó también un diagrama de

secuencia con la interacción general con la web para la visualización de un algoritmo.

Sprint 15

Corresponde con el periodo temporal del 12 al 19 de abril de 2023.

Durante este Sprint tuvo lugar el periodo de exámenes parciales de las asignaturas cuatrimestrales, y junto con las prácticas en empresa, no se pudo mantener la prioridad al desarrollo del proyecto. Por lo tanto, no existen puntos concretos a desarrollar, pero la idea era realizar, en la medida de lo posible, retoques en el mismo.

Los *violinplots* que se añadieron para la comparativa de algoritmos resultaban algo confusos. Aunque este tipo de gráficos se utiliza mucho para comparar datos, para este ámbito concreto no resultó ser la mejor opción. La parte superior de estos gráficos, que no son el máximo de los datos, sobrepasa en algunos casos el valor máximo de las estadísticas. Esto hace parecer que los cálculos son incorrectos. Se volvió a incluir los gráficos de caja anteriores.

Como no resultaba una tarea grande, se incluyó la integración continua a partir de este momento. La herramienta SonarCloud proporciona muchas métricas de calidad del código (*bugs*, línea duplicadas, seguridad...). Gracias a estos análisis se realizaron todas las posibles modificaciones para reducir las alertas que hasta el momento podían abordarse. Por ejemplo, la protección contra CSRF (*Cross-Site Request Forgery*) no se había contemplado, pero la herramienta sí lo considera como un elemento prioritario.

Sprint 16

Corresponde con el periodo temporal del 19 al 26 de abril de 2023.

Puntos a desarrollar:

1. Añadir usuarios (con *login* y registro)
2. Crear un espacio personal para los usuarios
3. Continuar con la memoria
4. Estandarización en la configuración

Durante este sprint se iniciaron las tareas para mantener usuarios en la aplicación. El primer paso para realizar esto fue la creación del modelo de usuario en la base de datos. El usuario es simple, con un identificador, nombre, email y contraseña (este no es el objetivo principal de la aplicación).

A continuación, se crearon los pares de **endpoint** y plantilla para inicio de sesión y registro. Como este tipo de formulario requiere de una validación exhaustiva y que permita dar al usuario ayuda contextual, se utilizó Flask-WTForms para ello porque tanto desde el navegador como desde el **backend** permite esta validación. Esto trata los formularios como objetos, que incluso cada uno de ellos mantiene una lista de errores. En ambos **endpoints**, se realizan las típicas comprobaciones: al registrar que no exista ya una cuenta con el mismo email, en el inicio de sesión que la contraseña coincida con la almacenada, que exista el email... Respecto a las contraseñas, solo se almacena el Hash (SHA256) por lo que no se podría obtener la cadena real de vuelta (de forma sencilla). Y como se comentaba, cuando se detecta un error de este estilo, se muestra un mensaje (en rojo) con lo que ha ocurrido.

Con todo ello, se modificó la barra de navegación incluyendo el menú de los usuarios: registrarse/iniciar sesión y una vez el usuario ha iniciado sesión, un menú desplegable que le llevaría a su espacio personal y perfil.

Antes de continuar con el desarrollo, se incluyó un *mockup* con lo que se tenía pensado hacer.

Para que el usuario pudiera modificar sus datos, se creó otro pequeño formulario muy parecido al de registro. Por su puesto, con su correspondiente ruta (**/perfil**) que valida que los datos introducidos son correctos.

Del mismo modo, era interesante que el usuario pudiera ver algo de su actividad, la idea era guardar los conjuntos de datos que ha subido para poder volver a utilizarlos y un historial de ejecuciones con el objetivo de replicar exactamente esa ejecución. Como se vio que la ruta del perfil y esta nueva eran parecidas, se unificó el estilo de tal forma que en la columna de la izquierda apareciesen los principales datos del usuario y a la derecha, o bien la modificación del perfil (**/perfil**) o su actividad (**/miespacio**).

Antes de seguir, para poder almacenar toda esta actividad, se crearon dos nuevas entidades «Dataset» y «Run» que almacenarían la información principal como los nombres de los ficheros, fecha de subida/ejecución...

La parte visual de la actividad se realizó con DataTables, un plug-in de jQuery que proporciona mucha versatilidad para hacer tablas. Los datos no provenían de las plantillas, se crearon nuevas rutas a las que hacer peticiones (como una API). Una vez que se había trasteado con este *plug-in* e incluso

vista la forma en la que eliminar filas, la idea es realizar una petición y que los datos recibidos sean incrustados en estas tablas. La obtención de los datos no fue un problema, se utilizaron los «*fetch*» de JavaScript y cuando se obtenía respuesta, se creaban las tablas. El verdadero problema fue la incorporación de acciones (como la de borrar), en la propia documentación de DataTables y gracias a su foro, se consiguieron generar botones en esta columna (con puro HTML). El siguiente reto fue crear otras peticiones como respuesta a los clics de estos botones. La suerte fue que DataTables permite obtener los datos de una fila directamente, lo que resultó de mucha ayuda para, por ejemplo, extraer el nombre de un fichero y posteriormente incorporarlo a las peticiones de borrado. Se añadieron también unos *modales* para las comprobaciones del usuario.

El desarrollo anterior se ha simplificado, primero se creaban versiones sencillas y luego se iban incorporando retoques hasta el resultado final. Por ejemplo, al final del todo, se añadió el estilo «*responsive*» a estas tablas. La ventaja fue que DataTables ya tenía extensiones que realizaban esto automáticamente.

Continuando con la memoria, se añadieron los objetivos de proyecto y el comienzo de aspectos relevantes.

A la vez que se iban incorporando estas adiciones, no se perdía de vista el resto de la aplicación, en la fase de configuración de los algoritmos resultaba conveniente permitir al usuario estandarizar o no el conjunto de datos a mostrar (hasta ahora siempre se estandarizaba), se incluyó un interruptor para ello.

Además de todo esto, se hicieron algunos cambios pequeños: «Footer» al final de la página, selección de fuentes para la página, arreglar algunos *bugs* menores o estilar el «*card*» del perfil.

Sprint 17

Corresponde con el periodo temporal del 26 de abril al 3 de mayo de 2023.

Puntos a desarrollar:

1. Botón de idioma.
2. Panel de administración.
3. Iconos de Bootstrap.

4. Mensajes flash (alertas).
5. Actualizar técnicas y herramientas.

Lo primero que se hizo fue sustituir los SVG de los iconos que se estaban utilizando, por clases de *Bootstrap icons*. Era muy engorroso tener cadenas tan grandes de texto y esta modificación solo conlleva añadir un enlace.

Lo siguiente que se realizó fue añadir un botón para cambiar el idioma. La idea es muy sencilla, las traducciones ahora se realizan detectando qué idioma se ajusta más al navegador del usuario (en las peticiones existe un «*header*» que incluye esta información), como también se quiere que el usuario decida, todas las peticiones son procesadas previamente por un *endpoint* (Flask permite esto) que detecta si existe un parámetro «*lang*» (`?lang=X`) y si lo hay, establece el idioma a este (guardándolo en la sesión). De esta forma se consigue no modificar lo ya existente.

El panel de administración fue un verdadero reto, como no se quería tener que hacer todo de cero (mucho código duplicado), se tuvo que pensar primero la forma de aprovechar las tablas que ya se habían creado. El panel de administración solo iba a tener una tabla nueva, además de la de los ficheros y ejecuciones que ya existían. Estas últimas son las que se debían reutilizar. Para ello, la información que debía ver el administrador, además de lo que ya se mostraba, era el usuario al que pertenecen los ficheros y ejecuciones. Se añadió esta columna a las tablas y las funciones se parametrizaron con un «*flag*» que permitiese especificar si la tabla debe generarse como administrador o como usuario normal. En el caso del panel de administración, mostrar esa columna de usuarios. Esto llevó a modificar también los modelos pues hasta ahora solo se almacenaba una clave foránea con el identificador del usuario (había que añadir el email) con una relación uno a varios.

Finalmente, se creó la tabla de usuarios del mismo modo que el resto, creando una ruta donde consultar todos los datos y añadir las acciones correspondientes de editar el usuario o eliminarlo. Se reutilizó mucho de lo que ya se había codificado. La ventaja es que, aunque se tuvieron que añadir bastantes cosas, ya se tenía el conocimiento previo así que todo fue más laborioso (el desarrollo anterior se dividió en varios días) pero menos difícil.

Además de lo comentado anteriormente, se quiso incluir unas estadísticas sencillas en cada una de las tablas. Para los usuarios, el total y para los ficheros subidos y ejecuciones, el total de los últimos siete días.

Finalmente se tuvo que modificar algún *endpoint* para aumentar las comprobaciones de seguridad y sobre todo, para evitar duplicar código. Por ejemplo, la edición de un usuario por parte de un administrador se «delega» al propio *endpoint* del perfil. En esta línea, a medida que pasaba el sprint, mejoraban las comprobaciones y controles de errores y para que el usuario tuviera algo de información, se añadió un modal que se activa cuando ocurre un error.

Los mensajes flash se modificaron como alertas de Bootstrap de tal forma que el color de las mismas dependiera de las categorías (se añadieron estas a todos los puntos en los que se lanzaba un mensaje flash).

Se actualizó el apartado de las técnicas y herramientas hasta este punto.

Además, también se realizaron otras modificaciones:

- Se añadió el botón de eliminación de ejecuciones, pues al igual que usuarios y ficheros, era interesante que pudieran eliminarse las ejecuciones.
- Se corrigió un error en la edición de perfil en el que si la contraseña actual se introducía correcta, ocurría un error crítico (no se habían pasado todas las variables a la plantilla porque el código no estaba actualizado). Se actualizó el *endpoint* correspondiente con todas las variables necesarias.
- Se corrigió un error en la reducción de dimensionalidad en el que cuando se desactivaba PCA y se incluía la misma columna en CX y CY, ocurría una excepción. Pandas parece no permitir concatenar una columna a un DataFrame que tiene columnas idénticas. Cuando ocurre esta situación, solo se incluye una columna.
- Se hicieron varias refactorizaciones generales para evitar código duplicado.

Sprint 18

Corresponde con el periodo temporal del 3 al 10 de mayo de 2023.

Puntos a desarrollar:

1. Añadir «*Toasts*» como alertas.
2. Arreglar visualizaciones (*tooltips*).

3. Cambiar formularios a WTForms.
4. Mostrar parámetros de las ejecuciones en el historial.
5. Estudiar y añadir conceptos teóricos Tri-Training.
6. Estudiar y añadir conceptos teóricos HTTP.

En primer lugar, el aspecto que se le había dado a los mensajes flash de la aplicación parecía un poco pobre. Bootstrap tiene los llamados «Toasts» que son avisos más sofisticados y estilados. Partiendo un ejemplo de la documentación, simplemente se filtra por la categoría del mensaje (error, advertencia, información...) y se genera el «*Toast*» con unos colores acordes.

Durante el Sprint anterior se estuvieron probando las visualizaciones. El problema es que los `tooltips` mostraban información confusa.

En primer lugar, estos recuadros que aparecen al pasar el ratón por un punto, no se posicionaban correctamente. Mediante un ejemplo encontrado en Internet², se vio que el propio D3 tiene una función que determina la posición del ratón correctamente. Además, el elemento `tooltip` tenía una posición absoluta respecto de toda la página para arreglarlo, se le añadió la posición relativa al contenedor. Esto se aplicó a los cuatro algoritmos.

El mayor problema era arreglar la información confusa. Como es obvio, el conjunto de datos que introduzca el usuario puede tener puntos duplicados y además al realizar PCA, puede ocurrir que dos datos aparezcan en la misma posición. Esto no se estaba contemplando y no se indicaba. Es aún más confuso en *Democratic Co-Learning* y *Tri-Training* pues cada punto puede ser clasificado por varios clasificadores e incluso en varias iteraciones.

Para ayudar al usuario a tener toda la información, el `tooltip` muestra ahora los puntos duplicados. Es decir, cuando solo hay un punto, simplemente mostrará la información normal, pero cuando haya puntos solapados (quizá por PCA o porque en el propio conjunto hay datos duplicados), los detectará y mostrará una lista de los identificadores de los puntos duplicados.

Para determinar esto lo que se ha hecho es comprobar (mediante diccionarios) si un punto ya había sido visto, en cuyo caso existen puntos solapados.

²Ejemplo de `tooltip`: <https://observablehq.com/@clhenrick/tooltip-d3-convention>

Otra cosa añadida al *tooltip* es la iteración de clasificación. Cuando el punto se acaba de clasificar o estamos en una iteración posterior, al lado de la etiqueta se muestra entre paréntesis la iteración en la que se clasificó.

Al final del sprint (y justo coincidiendo con una reunión de seguimiento), se planteó que el *tooltip* podía ser simplificado. Por una parte, cada vez que se muestra todos los puntos, cada uno de ellos tiene su posición X e Y. Ahora solo se muestra como si fuera un título una única vez.

Otra idea era que no se mostrase información futura (para *Democratic Co-Learning* y *Tri-Training*). Por ejemplo, si dos clasificadores habían etiquetado un punto en iteraciones 4 y 5 respectivamente, hasta ahora, en la iteración 2 (por ejemplo), el *tooltip* tenía «reservado» el espacio para escribir esos dos etiquetados. El comportamiento bueno sería que en una iteración menor que 4 simplemente mostrase un único texto para ese punto (ejemplificando que todavía no ha sido etiquetado). Cuando se llega a la iteración 4, se mostrará la etiqueta que le dio el primer clasificador, pero sin mostrar todavía el de la iteración 5. Y así sucesivamente.

En cuanto a los formularios, los de configuración del algoritmo estaban hechos directamente en HTML. Dado que se había empezado a utilizar WTForms (inicio de sesión, registro...), era conveniente modificarlos y trabajar con esta biblioteca. Todos los formularios tienen una parte común, lo primero que se hizo es crear una clase de formulario base del que el resto extendería. A partir de aquí, simplemente se trasladó todo lo que se tenía en HTML a los distintos *Fields* de WTForms. En el HTML simplemente se hace referencia a estos campos. Fue un proceso laborioso aunque sencillo.

Esto además permitía añadir el «token» CSRF para arreglar ciertas vulnerabilidades.

En la tabla del historial de ejecuciones faltaba la posibilidad de ver los parámetros de las ejecuciones. Para verlos, se añadió un campo de texto en la entidad de la base de datos de ejecuciones.

El formato JSON se pensó que era suficientemente legible como para mostrar los parámetros con él. Para ello, se creó una función (en Python) que transforma todos los parámetros que provienen del formulario de configuración en un diccionario. Este diccionario luego es transformado en texto para almacenarlo en ese campo nuevo.

Desde la web, simplemente se lee ese campo de la ejecución y se formatea como JSON. Existe un nuevo botón en la tabla que permite mostrar un «modal» con los parámetros.

Se añadieron los conceptos teóricos de HTTP, con una descripción del protocolo, la estructura de las peticiones, **cookies** y el ataque CSRF (*Cross-Site Request Forgery*) junto a su solución («*CSRF token*» comentado anteriormente).

En cuanto a *Tri-Training*, se añadió el pseudocódigo del mismo.

Además, también se realizaron otras modificaciones, las más notorias:

- Añadir favicon.
- Hacer el control de errores.
- Arreglar bugs en las tablas del espacio personal.
- Añadir algún aspecto relevante a la memoria.

Sprint 19

Corresponde con el periodo temporal del 10 al 17 de mayo de 2023.

Puntos a desarrollar:

1. Anexo requisitos.
2. Anexo documentación de usuario.
3. Actualizar manual del programador.
4. Diseño arquitectónico.
5. Técnicas y herramientas
6. Conceptos teóricos del tratamiento de datos.

En este Sprint se quería aumentar bastante la documentación. El primer objetivo era realizar el anexo de los requisitos. Lo primero fue añadir el catálogo de requisitos que la aplicación debía realizar.

A continuación, se realizó un primer diagrama de casos de uso general. A partir de este, se realizó la descripción de todos ellos (con sus pasos, precondiciones, excepciones, requisitos asociados...). Sin embargo, durante la realización de estos, se vio que el diagrama podía ser simplificado. El administrador, al igual que los usuarios, pueden eliminar los ficheros y ejecuciones, y por esta razón realizan el mismo caso de uso en ambos casos.

Se actualizó el diagrama de casos de uso y se añadieron todas las descripciones acordes de todos ellos.

El anexo de la documentación de usuario se podría realizar más adelante por si la aplicación sufre algún cambio final. Sin embargo, estos cambios serán, probablemente, en cuanto aspecto. Es por esto que las instrucciones útiles para el usuario serán las mismas. Además, resultaba una tarea apetecible (más motivadora) en esos momentos. En este manual se explicó todo lo que un usuario podía realizar (anónimo o registrado). Además, como también existe un rol administrador, se describieron todos sus privilegios.

En el anexo del diseño se añadió el diseño arquitectónico que se ha planteado para la aplicación, una arquitectura de tres capas. Se explicaron los conceptos relativos junto a cómo se reflejaba en la aplicación global.

Se añadió PEP8 a las técnicas y `Draw.io`, `Putty` y `Bootstrap Icons` en las herramientas.

Y por último, como otro gran apartado de este sprint, se añadieron algunos conceptos que se han utilizado en la aplicación sobre el tratamiento de datos. Pese a que la aplicación no realiza un pre-procesado completo, convenía explicar la codificación de variables categóricas, el análisis de componentes principales (PCA) y la estandarización que se puede aplicar a los datos al visualizarlos.

Además, también se realizaron numerosos cambios:

- Se eliminó el efecto «hover» en la barra de navegación que causaba muchos problemas en móviles.
- Se mejoró la visualización de los parámetros de las ejecuciones para que fuera más legible (nombre apropiados, no los identificadores de los formularios) y con traducciones acordes.
- Hacer el control de errores en la visualización.
- Se actualizó el pseudocódigo del *Tri-Training* y se añadió a la página web. (actualizado y añadido a Web).
- Se fijaron dependencias del proyecto.
- Se mejoró la comprobación de las extensiones de ficheros para que obligatoriamente terminen en «.arff» o «.csv».
- Se añadió el diagrama de despliegue en el diseño.

Sprint 19

Corresponde con el periodo temporal del 17 al 24 de mayo de 2023.

Puntos a desarrollar:

1. Análisis de viabilidad: económica y legal.
2. Añadir Scrum como técnica.

El desarrollo se vio ralentizado en este sprint debido al periodo de convocatoria final de evaluación.

Para ir completando los anexos se añadió tanto la viabilidad legal como económica. En el caso de la económica se contaba con algún ejemplo de cómo se debía realizar de alumnos previos (proporcionado por los tutores). Durante su desarrollo se contemplaron todos los costes derivados (suposiciones) de los empleados y de activos *hardware* y *software*. Además, se tomó la decisión lógica de no incluir beneficios, ya que se trata de una aplicación completamente docente y pensada para que cualquiera pueda usarla.

En cuanto a la viabilidad legal, se estudiaron todas las licencias del *software* que se ha empleado para determinar cuál sería la más apropiada. Finalmente se escogió la licencia *BSD 3-Clause*.

Como la metodología Scrum es altamente utilizada (y ya no resulta una clara novedad), se incluyó en las técnicas una pequeña explicación de todo lo que interviene en ella.

Sprint 20

Corresponde con el periodo temporal del 24 al 31 de mayo de 2023.

Puntos a desarrollar:

1. Añadir la validación de los algoritmos.
2. Terminar conceptos teóricos.
3. Añadir la descripción de los formatos de *tooltip*.
4. Añadir pruebas del sistema.
5. Hacer una revisión general de la aplicación, memoria y anexos.

En sprint lo que planificó era tener todo el proyecto lo más cerca posible al objetivo para que una vez se realice la reunión del sprint evaluar cuál es el estado.

En primer lugar, algo que se estaba retrasando constantemente debido a la prioridad de nueva funcionalidad era la validación de los algoritmos. En este sentido sí que se tenían bastante probados, pero faltaba añadirlo a la documentación.

Como se venía comentado en sprints anteriores, esta validación se trata de comparar la implementación propia contra la de *sslearn* mediante gráficos de cajas. La idea es que si se obtienen resultados similares en las estadísticas comparadas, se puede concluir que la implementación propia es *correcta*. El resultado final con la ejecución de cada algoritmo con varios conjuntos de datos fue positivo y se concluyó que eran correctos.

Un punto a destacar de la validación fue la diferencia de implementación de Co-Training. En líneas generales, lo que se pudo observar de *sslearn* es que el tamaño de *pool* era más grande que lo que se venía realizando. Esto se subsanó aumentando los parámetros en la llamada al algoritmo propio para hacerlo lo más parejo posible. Aunque sigue apareciendo mayor variación que el resto de los algoritmos, se puede comparar perfectamente y llegar a la misma conclusión.

En cuanto a los conceptos teóricos, se añadió la descripción de *Tri-Training* (el pseudocódigo ya estaba incluido). Para ello se revisó de nuevo el artículo correspondiente para incluir la idea general más importante.

Algo que podía ser muy confuso para un usuario nuevo es la multitud de formatos de *tooltip* de la visualización. Se añadió la explicación de cada uno para cada algoritmo en el manual de usuario.

En cuanto a los anexos (a expensas de revisar y perfeccionar), el apartado que faltaba era el de las pruebas del sistema en el manual del programador. Por sugerencia de los tutores, se ha utilizado Selenium como entorno de pruebas. Además, en la asignatura de «Validación y Pruebas» se utilizó Katalon Recorder, que utiliza Selenium. Esto facilitó la parte de las pruebas automáticas.

La idea de las pruebas era tener los suficientes casos de prueba que asegurasen el funcionamiento de la aplicación. De hecho, durante algunos cambios que se hacían, se volvían a ejecutar para comprobar que esos cambios no rompían la aplicación. En total fueron 17 casos de prueba teniendo en cuenta las visualizaciones, los usuarios (y sus permisos) y las acciones de estos (incluidas las del administrador).

Durante todo el sprint se intentaba corregir todo lo que se veía como mejora (revisión general). Entre estas correcciones están:

- Arreglar el posicionamiento de los *tooltips* de ayuda contextual (no los de la visualización) sustituyéndolos por los de bootstrap.
- Arreglar adaptabilidad en el panel de administración ya que las tablas de las pestañas dos y tres no se ajustaban (solo la primera).
- Anonimizar las rutas de usuario, en el sentido de que hasta ahora se incluía por ejemplo el identificador del usuario en la base de datos (algo como `/perfil/2`). Convenía no mostrar estos valores (`/perfil`).
- Añadir traducciones.
- Justificar el por qué de la licencia BSD 3-Clause.

Sprint 21

Corresponde con el último periodo temporal (considerando hasta la entrega).

Puntos a desarrollar:

1. Enlazar el manual en web.
2. Logos de los algoritmos.
3. Depurar aplicación.
4. Completar memoria (resumen, conclusiones y líneas futuras).
5. Añadir videotutoriales.
6. Completar anexos.

En principio, como sprint final, no se planteaba realizar ninguna implementación. Quizá alguna modificación/mejora que no altere el funcionamiento.

En primer lugar, se añadió en la barra de navegación un enlace directo al manual de usuario de la propia documentación del proyecto. Esto conllevó la actualización de las capturas del manual para que se apreciara el nuevo icono y un nuevo caso de uso sencillo.

En la aplicación web faltaban los logos de los algoritmos. Para ello, se generó un modelo de un robot en Dall-e³ (consejo de los tutores). A partir de aquí, se fue modificando el logo base adecuándolo al contexto de los algoritmos.

- Self-Training: Único robot.
- Co-Training: Son dos clasificadores (dos robots) pero están combinados (funcionamiento interno del algoritmo), ya que es como si actuaran como uno solo, pero cada uno ve una parte de los datos (*multi-view*).
- Democratic Co-Learning: Tres robots con su papeleta de voto (se basa en la votación).
- Tri-Training: Se combina el conocimiento (predicciones) de dos clasificadores para que un tercero aprenda.

En cuanto a la depuración de la aplicación lo que se intentó fue mejorar la interacción del usuario. En este sentido, se modificaron los formularios de configuración de Self-Training y Co-Training.

En Self-Training el límite o *threshold* ya no se realiza con un *placeholder*, sino con un rango de porcentaje, más sencillo de usar. Además, se ha incluido un *tooltip* indicando que si introduce 0 en el número de iteraciones no se limita internamente (0 es como infinito).

En Co-Training se ha añadido el *tooltip* del número de iteraciones como Self-Training (al poner 0 es como si fuera infinito). También se ha añadido control del formulario para impedir que el número de positivos y negativos sean nulos a la vez.

Tanto en Democratic Co-Learning y Tri-Training se ha incluido la comprobación de diversidad en los clasificadores (deben ser diversos, diferentes). Se muestra un mensaje en el caso de que no sean distintos. De hecho, al entrar en la configuración, aparecen seleccionados tres distintos.

Entre otras mejoras, en el gráfico de estadísticas, cuando había muchas iteraciones, los números se solapaban. Se realiza una operación de división entre 35 (a partir de este los números se solapaban) y ese resultado actúa de divisor en el número de «rayitas» dibujadas en las iteraciones. De esta forma, se evitan los solapamientos, pero se mantienen suficientes marcas.

³<https://openai.com/dall-e-2>

Durante unos días se sustituyó los enlaces «CDN» (para descargar los recursos o bibliotecas) por ficheros locales descargados. Esto fue revertido pues convenía delegar estas descargas al usuario (no sobrecargar el servidor con tráfico innecesario).

Como último punto a destacar (además de otros retoques menores), se incorporó la expiración de las *cookies* de sesión. Cuando el usuario cierra el navegador, esta *cookie* es eliminada (al volver a la aplicación será como la primera vez).

Al mismo tiempo que se fue depurando la aplicación se iba finalizando con la memoria. Se añadieron las conclusiones derivadas, el resumen del proyecto y las líneas futuras para la expansión de la aplicación.

Además, tanto para la memoria como los anexos se fueron corrigiendo los errores vistos por ambos tutores.

Para los anexos, también se fueron incorporando los elementos que faltaban (como el diagrama de componentes) y completando aquellas partes con alguna explicación más o aclaración. Por último, se añadieron unos videotutoriales (enlazados a esta documentación) para ejemplificar las principales acciones de los usuarios (anónimos, registrados y administradores).

Finalmente, surgió la posibilidad de añadir las estadísticas específicas a Co-Training, ya que al ser dos clasificadores, podría aportar alguna idea sobre el comportamiento global. Se añadió sin problemas debido a que la implementación estaba preparada para funcionar de forma genérica (mismas funciones utilizadas en Democratic Co-Learning y Tri-Training).

Y en definitiva, aquí concluye el desarrollo proyecto. Ha sido un largo camino e intenso, pero gratificante y con mucho aprendizaje.

A.3. Estudio de viabilidad

Viabilidad económica

El estudio de la viabilidad económica viene a evaluar cuáles son los costes derivados del desarrollo del proyecto y beneficios esperados.

Aún con todo esto, el contexto del proyecto, desde el inicio, se ha marcado como una aplicación de tipo docente. Esto tendrá ciertas implicaciones que se irán comentando durante este estudio.

Se supondrá un entorno real de una empresa creada con el fin de llevar a cabo este proyecto. Esta empresa contará con un empleado desarrollador (alumno), dos supervisores (tutores) y los activos hardware y software. El comienzo oficial del proyecto fue el 25 de enero de 2023 y su fecha límite estipulada es el 8 de junio (unos 5 meses).

Costes

Empleados:

El desarrollador ha realizado una media de 4.5 horas diarias (progresivamente aumentando hasta la fecha límite de entrega): unas 3 horas en días laborables y en los fines de semana se empleaban alrededor de las 8 horas.

El cómputo del salario bruto del alumno, suponiendo un salario de 11€/hora, según lo estipulado en las bases mínimas de Ingenieros y Licenciados en [4] es:

$$4,5 \frac{\text{horas}}{\text{día}} \times 7 \frac{\text{días}}{\text{semana}} \times 11 \frac{\text{€}}{\text{hora}} \times 4 \frac{\text{semanas}}{\text{mes}} = 1\,386 \frac{\text{€}}{\text{mes}}$$

Para el caso de los supervisores (tutores), el tiempo empleado de ambos es de 2 horas/semana para la revisión del progreso y reuniones. Por lo que, suponiendo mismas bases mínimas anteriores, y añadiendo cuatro euros debido al cargo superior, el salario bruto es:

$$2 \frac{\text{horas}}{\text{semana}} \times 15 \frac{\text{€}}{\text{hora}} \times 4 \frac{\text{semanas}}{\text{mes}} = 120 \frac{\text{€}}{\text{mes}}$$

Constituido el salario bruto que recibe cada empleado, deben añadirse todos los impuestos de la Seguridad Social estipulados para el año 2023. Estos impuestos están recogidos en la web oficial de la seguridad social en el «Régimen General de la Seguridad Social» en [4].

En la tabla A.1 se puede ver un resumen de los impuestos que han de aplicarse al salario bruto de los empleados considerando categorías generales.

Concepto	Impuesto(%)
CONTINGENCIAS (Comunes)	23,60
DESEMPLEO (Tipo General)	5,50
FOGASA	0,20
FORMACIÓN PROFESIONAL	0,60

Tabla A.1: Tabla de impuestos

Por lo que la fórmula del cómputo total que la empresa debe pagar por los empleados es:

$$\text{Gasto de la empresa} = \frac{\text{Salario}}{1-(0,236+0,055+0,002+0,006)}$$

Figura A.5: Cálculo del gasto de la empresa por empleado

Empleado	Salario bruto (€)	Gasto de la empresa (€)
David Martínez Acha (des)	1 386	1 977,18
Álvar Arnaiz González (sup)	120	171,18
César Ignacio García Osorio (sup)	120	171,18
Total	1 626	2 319,54
Total 5 meses	8 130	11 597,7

des: desarrollador
sup: supervisor

Tabla A.2: Salarios brutos y coste que supone a la empresa

En la tabla A.2 se presentan todos los costes de los empleados. El total, para los 5 meses de duración aproximada es 11 597,70€.

Hardware:

Para desarrollar el proyecto se ha utilizado un ordenador de sobremesa montado por piezas. Está compuesto por un AMD Ryzen 7 5800X de 8 núcleos a 3.8 GHz, 16 GB de memoria RAM y 1TB SSD NVMe M.2. Este equipo está valorado en 1 800€. La vida útil de este activo es de 3 años.

Además, para hacer la aplicación accesible a todo el mundo, se ha adquirido un servidor para ello (supuesto considerado). Este servidor está compuesto por dos AMD EPYC, 8 GB de memoria RAM y 80gb de SSD. El equipo está valorado en 1 300€. La vida útil de este activo es de 3 años.

Teniendo en cuenta que la duración del proyecto es de 5 meses, el cálculo de la amortización se realiza de la siguiente forma:

$$\text{Amortización} = \frac{\text{Coste}}{\text{Vida útil}} \times \frac{5}{12}$$

Figura A.6: Cálculo de la amortización para este proyecto

En la tabla A.3 se presentan los costes de estos activos y el coste amortizado de los mismos.

Activo	Coste (€)	Coste amortizado (€)
Equipo sobremesa	1 800	250
Servidor	1 300	180,55
Total	3 100	430,55

Tabla A.3: Costes del hardware

Software:

Para el desarrollo del producto se ha hecho uso de *software*. En la medida de lo posible se han utilizado licencias gratuitas y/o proporcionadas por pertenecer a la Universidad de Burgos.

Los activos software que han supuesto un coste para la empresa es la adquisición del sistema operativo Windows 10 Pro del equipo de sobremesa. Está valorado en 250€. Para el cálculo de la vida útil, se tiene en cuenta la finalización del soporte de Microsoft el 14 de octubre de 2025, aproximadamente dos años y medio.

El coste amortizado será 41,66€ (aplicando el mismo cálculo que antes).

Costes indirectos:

Se consideran unos gastos fijos indirectos iguales al 12 % del resto de gastos anteriores. A estos se le añade otros gastos considerados.

Concepto	Coste (€)
Dominio <code>dmacha.dev</code>	12,00
12 % costes	1 448,39
Total	1 460,39

Tabla A.4: Costes indirectos

Total:

En la tabla A.5 se presenta la suma de todos los costes descritos.

Tipo de coste	Coste (€)
Empleados	11 597,70
Hardware	430,55
Software	41,66
Indirectos	1 460,39
Total	13 530,30

Tabla A.5: Coste total

El coste total del proyecto, finalmente, es de 13 530,30€.

Beneficios

La aplicación no está pensada para extraer beneficio con su uso. Se trata de una aplicación de índole docente que permite el acceso completo a todos los usuarios. Esto incluye las ventajas que un usuario registrado tiene sobre uno anónimo, este último puede adquirir dichas ventajas de forma gratuita (registro).

Viabilidad legal

El estudio de la viabilidad legal se basa en las licencias de las bibliotecas/librerías que se han utilizado en el desarrollo. A partir del listado de estas, se estudiará la permisividad en conjunto para saber cuál es la más restrictiva y seleccionar acordemente la de este proyecto.

Librería	Versión	Licencia
<i>Pandas</i>	1.4.3	BSD 3
<i>Babel</i>	2.11.0	BSD 3
<i>Flask</i>	2.2.5	BSD 3
<i>Flask-babel</i>	3.0.1	BSD 3
<i>Flask-SQLAlchemy</i>	3.0.3	BSD 3
<i>Flask-Login</i>	0.6.2	MIT
<i>Scikit-learn</i>	1.2.0	BSD 3
<i>Numpy</i>	1.23.3	BSD 3
<i>Scipy</i>	1.9.3	BSD 3
<i>Setuptools</i>	65.5.1	MIT
<i>Pytest</i>	7.2.0	MIT
<i>Werkzeug</i>	2.2.3	BSD 3
<i>Matplotlib</i>	3.7.1	PSF ⁴
<i>WTForms</i>	3.0.1	BSD 3
<i>Flask-WTF</i>	1.1.1	BSD 3
<i>SQLAlchemy</i>	2.0.13	MIT
<i>Sslearn</i>	1.0.3.1	BSD 3
<i>Imbalanced-learn</i>	0.10.1	MIT
<i>Gunicorn</i>	20.1.0	MIT
<i>Bootstrap</i>	5.2.3	MIT
<i>D3js</i>	7.8.4	ISC ⁵
<i>jQuery</i>	3.6.4	MIT
<i>DataTables</i>	1.13.4	MIT

Tabla A.6: Bibliotecas usadas, versiones y licencias

⁴Python Software Foundation License. A efectos prácticos solo conlleva realizar un pequeño resumen de lo que se ha modificado (y solo en caso de modificación/adición) [1].

⁵Internet Systems Consortium, equivalente a MIT.

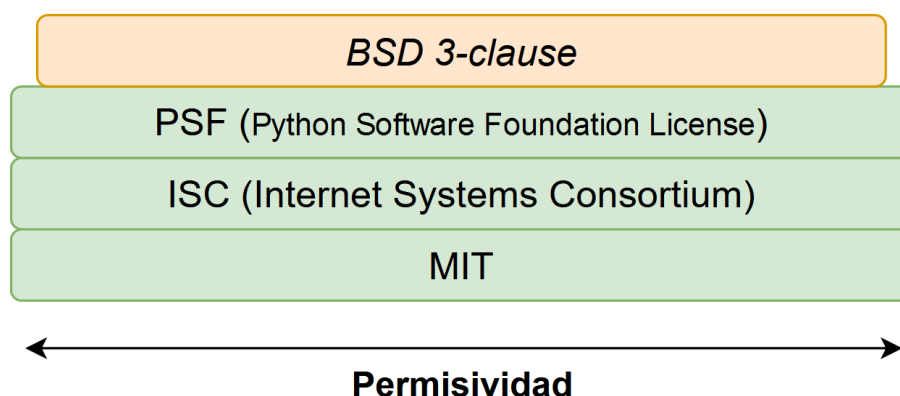


Figura A.7: Permisividad de las licencias.

Para tener cierto contexto, en la figura [A.7](#) se representan las licencias organizadas según las restricciones que imponen. Una anchura mayor representa mayores libertades. Se debe tener en cuenta que solo es una representación para determinar la licencia más restrictiva, es decir, no es una representación calculada con exactitud.

La peculiaridad de *BSD 3-Clause* es que incluye la cláusula de «no respaldo». Esta cláusula impide promocionar un producto con el nombre de la organización o de sus colaboradores (que han creado el software usado). Es por ello que su permisividad se ha considerado algo menor.

Selección Para seleccionar la licencia apropiada a este proyecto se tiene en cuenta que incluir una licencia menos restrictiva que *BSD 3-Clause* podría incumplir alguna de las «cláusulas» que esta impone. Por tanto, para mantener esas normas y condiciones impuestas, el proyecto tendrá la licencia **BSD 3-Clause**.

Implicaciones de BSD 3-clause Esta licencia tiene las siguientes implicaciones (traducción no oficial) [\[6\]](#):

Copyright (c) 2023, David Martínez Acha

1. Las redistribuciones del código fuente deben conservar el aviso de derechos de autor anterior, esta lista de condiciones y el siguiente descargo de responsabilidad.

ESTE SOFTWARE ES PROPORCIONADO POR LOS TITULARES

DE LOS DERECHOS DE AUTOR Y LOS CONTRIBUYENTES <<TAL CUAL>> Y CUALQUIER GARANTÍA EXPLÍCITA O IMPLÍCITA, INCLUYENDO, ENTRE OTRAS, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZABILIDAD E IDONEIDAD PARA UN PROPÓSITO EN PARTICULAR. EN NINGÚN CASO EL TITULAR DE LOS DERECHOS DE AUTOR O LOS CONTRIBUYENTES SERÁN RESPONSABLES POR CUALQUIER DAÑO DIRECTO, INDIRECTO, INCIDENTAL, ESPECIAL, EJEMPLAR O CONSECUENTE (INCLUYENDO, ENTRE OTROS, LA ADQUISICIÓN DE BIENES O SERVICIOS SUSTITUTOS; PÉRDIDA DE USO, DATOS O BENEFICIOS; O INTERRUPCIÓN DEL NEGOCIO) CUALQUIER CAUSA Y SOBRE CUALQUIER TEORÍA DE RESPONSABILIDAD, YA SEA POR CONTRATO, RESPONSABILIDAD ESTRICTA O AGRAVIO (INCLUYENDO NEGLIGENCIA O DE OTRO TIPO) QUE SURJA DE CUALQUIER MANERA DEL USO DE ESTE SOFTWARE, INCLUSO SI SE ADVIERTE DE LA POSIBILIDAD DE DICHO DAÑO.

2. Las redistribuciones en forma binaria deben reproducir el aviso de derechos de autor anterior, esta lista de condiciones y el siguiente descargo de responsabilidad en la documentación y/u otros materiales provistos con la distribución.
3. Ni el nombre del titular de los derechos de autor, ni los nombres de sus colaboradores, pueden utilizarse para respaldar o promocionar productos derivados de este software sin el permiso previo por escrito.

En general, resulta una licencia adecuada que brinda muchas libertades y dado el enfoque docente, es lo que se busca.

Apéndice *B*

Especificación de Requisitos

B.1. Introducción

En esta sección se detalla la especificación de requisitos, en ella quedará registrado qué es lo que la aplicación debe hacer (y cómo).

B.2. Objetivos generales

Los objetivos del proyecto pueden resumirse en los siguientes:

1. Implementación de *Self-Training*, *Co-Training*, *Democratic Co-Learning* y *Tri-Training*.
2. Creación de una aplicación web e integración de los cuatro algoritmos en la aplicación web para su visualización.
3. Sistema de usuarios que les permita controlar sus ficheros y ejecuciones.
4. Administración de los usuarios y todos los ficheros y ejecuciones (mediante un rol administrador).

El núcleo del proyecto, pese a la importancia de las implementaciones los algoritmos, es la aplicación web, pues representa la parte interactiva del proyecto. Por este motivo, el presente apartado analizará con precisión qué es lo que la aplicación debe hacer (con sus limitaciones).

B.3. Usuarios de la aplicación

Como se ha comentado, existirá un sistema de usuarios en el que podrán diferenciarse: Usuario, Administrador y Anónimo (este último así lo denomina el sistema de gestión de cuentas utilizado, *Flask-Login*).

- **Anónimo:** Usuario que no posee una cuenta registrada en el sistema, es el usuario base. Puede utilizar toda la aplicación básica: seleccionar algoritmo, subir un conjunto de datos, configurar el algoritmo y visualizarlo.
- **Usuario:** Usuario que posee una cuenta registrada sin privilegios. Al igual que el anónimo base, puede utilizar la aplicación, pero además, los ficheros (conjuntos de datos) y ejecuciones son almacenados en el sistema.
- **Administrador:** Usuario que posee una cuenta registrada con privilegios. Es exactamente lo mismo que un usuario, con la particularidad de que podrá, además, gestionar estos usuarios, todos los ficheros y todas las ejecuciones (de todos los usuarios).

B.4. Catálogo de requisitos

A continuación se detallan los requisitos funcionales así como los no funcionales.

Requisitos funcionales

- **RF-1 Visualización de algoritmos semi-supervisados:** la aplicación debe permitir visualizar el proceso de entrenamiento de los algoritmos implementados (con sus estadísticas pertinentes).
 - **RF-1.1 Selección de algoritmo:** la aplicación debe permitir seleccionar uno de los algoritmos implementados.
 - **RF-1.2 Carga de conjunto de datos:** la aplicación debe permitir subir un fichero ARFF o ACSV con el conjunto de datos.
 - **RF-1.3 Configuración del algoritmo:** la aplicación debe permitir configurar la ejecución del algoritmo con los parámetros específicos del mismo.

- **RF-1.4 Control visualizaciones:** la aplicación debe mostrar visualizaciones interactivas: una principal (gráfico de dos dimensiones con los puntos del conjunto de datos) y una o varias estadísticas (gráficos de líneas).
 - **RF-1.3.1 Controlar paso (iteración):** la aplicación debe permitir controlar el paso del proceso de entrenamiento (iteración anterior/siguiente).
 - **RF-1.3.1 Interacción con el gráfico principal:** el gráfico principal debe mostrar información relevante en un *tooltip* cuando el usuario interactúe con cada punto.
- **RF-2 Manejo de usuarios:** la aplicación debe manejar cuentas de usuario.
 - **RF-2.1 Registro:** la aplicación debe permitir crear cuentas de usuario (no administrador).
 - **RF-2.2 Inicio de sesión:** la aplicación debe permitir el inicio de sesión a aquellos usuarios con cuenta.
- **RF-3 Personalización del perfil:** los usuarios con cuenta (no anónimos) deben poder modificar sus datos personales de su perfil de usuario.
- **RF-4 Espacio personal:** los usuarios con cuenta (no anónimos) deben poseer de un espacio personal.
 - **RF-4.1 Control de conjunto de datos:** los usuarios con cuenta deben poder consultar sus ficheros subidos.
 - **RF-4.1.1 Ejecución de un nuevo algoritmo:** los usuarios con cuenta deben poder utilizar esos ficheros en un algoritmo (nueva ejecución).
 - **RF-4.1.2 Eliminación:** los usuarios con cuenta deben poder eliminar esos ficheros del sistema.
 - **RF-4.2 Control de ejecuciones:** los usuarios con cuenta deben poder consultar sus ejecuciones anteriores (con toda su información).
 - **RF-4.2.1 Replicar ejecución:** los usuarios con cuenta deben poder replicar las ejecuciones.
 - **RF-4.2.2 Eliminación:** los usuarios con cuenta deben poder eliminar las ejecuciones.

- **RF-5 Administración:** la aplicación debe manejar cuentas de usuario de tipo administrador.
 - **RF-5.1 Control de usuarios:** el administrador debe poder consultar los usuarios registrados.
 - **RF-5.1.1 Edición de usuario:** el administrador debe poder modificar los datos de los usuarios.
 - **RF-5.1.2 Eliminación:** el administrador debe poder eliminar usuarios del sistema.
 - **RF-5.1.3 Total:** el administrador debe poder ver el número total de usuarios.
 - **RF-5.2 Control de conjuntos de datos global:** el administrador debe poder consultar todos los ficheros subidos.
 - **RF-5.2.1 Eliminación:** el administrador debe poder eliminar cualquier fichero de los usuarios registrados.
 - **RF-5.2.2 Últimos ficheros:** el administrador debe poder ver el **número** de ficheros subidos de los últimos 7 días.
 - **RF-5.3 Control de ejecuciones global:** el administrador debe poder consultar todas ejecuciones (con toda su información).
 - **RF-5.3.1 Eliminación:** el administrador debe poder eliminar cualquier ejecución de los usuarios registrados.
 - **RF-5.3.2 Últimas ejecuciones:** el administrador debe poder ver el **número** de ficheros subidos de los últimos 7 días.
- **RF-6 Cambio de idioma:** todos los usuarios deben poder cambiar el idioma (inglés o español).
- **RF-7 Acceso a la ayuda:** los usuarios deben tener a su alcance un manual de ayuda.

Requisitos no funcionales

Los requisitos anteriores definen de alguna manera lo que el sistema debe hacer. En este caso, los no funcionales son restricciones, por así decirlo, de calidad. Responden a la pregunta de cómo funciona y no qué es lo que hace. Todas estas restricciones son aplicadas de forma intrínseca en el desarrollo.

- **RNF-1 Disponibilidad:** el sistema de funcionar con muy alta probabilidad ante una petición. Es decir, debe encontrarse en condiciones de funcionamiento.

- **RNF-2 Accesibilidad:** el sistema debe poder abarcar el mayor público posible, facilitando su acceso y su manejo independientemente de las capacidades personales.
- **RNF-3 Soporte:** el sistema debe poder utilizarse en el mayor número posible de navegadores (dada su naturaleza Web).
- **RNF-4 Mantenibilidad:** el sistema debe ser fácil de modificar, mejorar o adaptar cuando se presenten nuevas necesidades.
- **RNF-5 Seguridad:** el sistema debe asegurar la información sensible (mediante cifrado y controles de accesos) y debe ser accesible mediante protocolos segurizados (SSL).
- **RNF-6 Privacidad¹:** el sistema debe respetar la información privada y, de ninguna forma, compartirla con terceros. Debe ser un espacio reservado confidencial con el usuario.
- **RNF-7 Escalabilidad:** el sistema debe ser capaz de crecer para ajustarse a la carga de trabajo.
- **RNF-8 Extensibilidad:** debe ser fácil añadir nueva funcionalidad al sistema, concretamente, la adición de nuevos algoritmos debe implicar pocas modificaciones.
- **RNF-9 Robustez:** el sistema debe ser altamente capaz de manejar los errores durante la ejecución (entradas erróneas, *bugs*...), mostrando información precisa al usuario y recuperándose de ellos a una situación estable.
- **RNF-10 Internacionalización:** el sistema debe estar preparado para soportar múltiples idiomas.

B.5. Especificación de requisitos

En la figura B.1 se detalla el diagrama de casos de uso general. Y a continuación, se detalla la descripción de cada uno de los casos de uso representados en el diagrama anterior.

¹La privacidad puede ser definida como el ámbito de la vida personal de un individuo, quien se desarrolla en un espacio reservado, el cual tiene como propósito principal mantenerse confidencial [8]

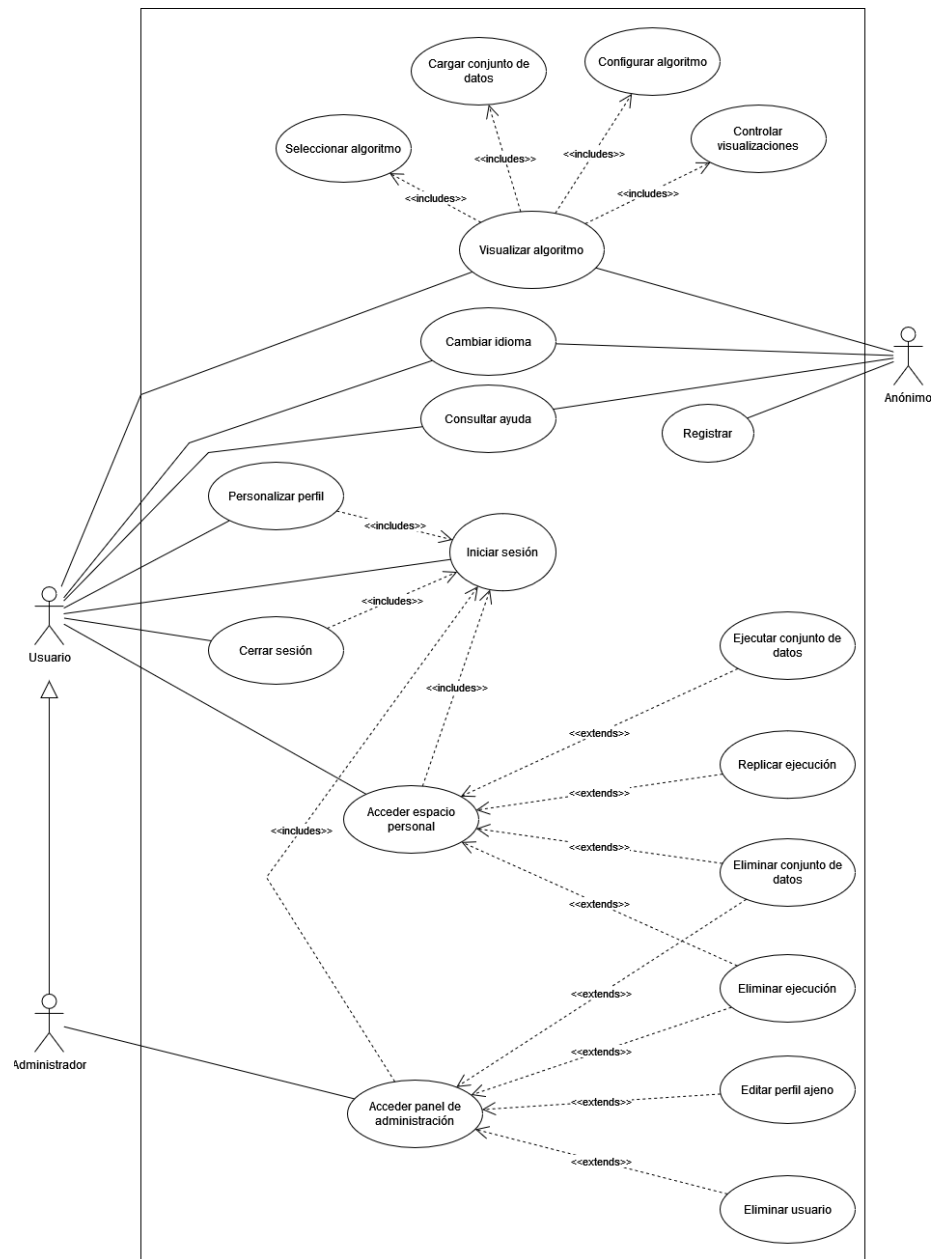


Figura B.1: Diagrama de casos de uso

CU-1	Visualizar un algoritmo
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-1.4
Descripción	Visualización del proceso de entrenamiento de un algoritmo semi-supervisado. Con gráfico principal y estadísticos.
Precondición	No hay precondiciones
Acciones	<ol style="list-style-type: none"> 1. Ejecución del caso de uso 2 (Seleccionar algoritmo). 2. Ejecución del caso de uso 3 (Cargar conjunto de datos). 3. Ejecución del caso de uso 4 (Configurar algoritmo). 4. Se realiza la ejecución interna del algoritmo, obtención de la información y creación de los gráficos. 5. El usuario verá en la página los distintos gráficos de su visualización. <p>Opcional Ejecución del caso de uso 5 (Controlar visualizaciones).</p>
Extensiones	4a Si el usuario estuviera registrado, toda la información de la ejecución será almacenada.
Postcondición	Visualizaciones renderizadas en la Web
Excepciones	<ul style="list-style-type: none"> ■ Excepciones de los casos de uso ejecutados controladas por ellos mismos. ■ El conjunto de datos tenía atributos nominales (paso 4). En este caso se mostrará un mensaje (modal) y al cerrarlo volverá al paso 3. ■ La ejecución del algoritmo finalizó con excepciones (paso 4), serán capturadas y se volverá al paso 3.
Importancia	Alta

Tabla B.1: Caso de uso 1: Visualizar un algoritmo.

CU-2	Seleccionar algoritmo
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-1, RF-1.1
Descripción	Seleccionar el algoritmo a ejecutar (establecerlo en la sesión del usuario).
Precondición	Ejecutando el caso de uso 1.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un algoritmo haciendo clic en el nombre del algoritmo en la barra de navegación. 2. Se redirige al usuario a la página de subida.
Acciones alternativas	<ol style="list-style-type: none"> 1. En caso de encontrarse en la página principal, el usuario puede seleccionar un algoritmo haciendo clic en una de las tarjetas de presentación de algoritmos. 2. Se redirige al usuario a la página de subida.
Postcondición	Algoritmo almacenado en su sesión y redirección a la página de subida.
Excepciones	Sin excepciones
Importancia	Alta

Tabla B.2: Caso de uso 2: Seleccionar algoritmo.

CU-3	Cargar conjunto de datos
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-1, RF-1.2
Descripción	Carga en el sistema de un fichero ARFF o CSV con el conjunto de datos a utilizar.
Precondición	Ejecutando el caso de uso 1 y haber ejecutado ya el caso de uso 2.
Acciones	<p>Opcional El usuario puede descargar un fichero de prueba si no lo tuviera haciendo clic en el botón de descarga.</p> <ol style="list-style-type: none"> 1. El usuario sube un fichero ARFF o CSV en la zona de subida (arrastrando o seleccionando del sistema). 2. Una vez que la carga se ha completado, se habilita el botón de configuración. 3. El usuario hace clic en dicho botón. 4. Se redirige al usuario a la página de configuración.
Extensiones	3a Si el usuario estuviera registrado, el fichero será vinculado a su cuenta (en base de datos).
Postcondición	Conjunto de datos almacenado en su sesión (y fichero local) y redirección a la página de configuración
Excepciones	<ul style="list-style-type: none"> ■ El usuario ha accedido a la página de subida sin seleccionar un algoritmo, será redirigido a la página principal (con un mensaje de aviso de esta situación) y se ejecutará el caso de uso 2. ■ El fichero no es ARFF ni CSV, al hacer clic en el botón, será redirigido a esta misma página (subida) y deberá realizar el caso de uso de nuevo.
Importancia	Alta

Tabla B.3: Caso de uso 3: Cargar conjunto de datos.

CU-4	Configurar algoritmo
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-1, RF-1.3
Descripción	Parametrización de la ejecución del algoritmo. Cada algoritmo tiene sus propios parámetros (mismo procedimiento).
Precondición	Ejecutando el caso de uso 1 y haber ejecutado ya el caso de uso 3.
Acciones	<ol style="list-style-type: none"> 1. El usuario verá el apartado teórico por un lado y el formulario de parámetros por otro. 2. El usuario selecciona e introduce los parámetros deseados para la ejecución del algoritmo. 3. El usuario hace clic en el botón de ejecución. 4. Se redirige al usuario a la página de visualización.
Postcondición	Redirección a la página de visualización
Excepciones	<ul style="list-style-type: none"> ■ El usuario ha accedido a la página de subida sin seleccionar un algoritmo, será redirigido a la página principal (con un mensaje de aviso de esta situación) y se ejecutará el caso de uso 2. ■ El usuario ha accedido a la página de configuración sin cargar un conjunto de datos, será redirigido a la página de subida (con un mensaje de aviso de esta situación) y se ejecutará el caso de uso 3. ■ El usuario sí que ha cargado un conjunto de datos, pero el sistema no puede acceder al fichero. Redirección a página de error, deberá volver a intentar el caso de uso. ■ El formulario está incompleto o erróneo. En este caso se bloqueará el envío del formulario indicando qué se debe arreglar.
Importancia	Alta

Tabla B.4: Caso de uso 4: Configurar algoritmo.

CU-5	Controlar visualizaciones
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-1, RF-1.4
Descripción	Manipulación de las visualizaciones de forma interactiva.
Precondición	Ejecutando el caso de uso 1 y haber ejecutado ya el caso de uso 4.
Acciones	<ol style="list-style-type: none"> 1. El usuario verá la visualización principal así como las estadísticas. 2. El usuario puede avanzar o retroceder en las iteraciones del algoritmo. 3. El usuario puede realizar <i>zoom</i> sobre el gráfico principal así como reiniciarlo. 4. El usuario puede ver información particular sobre cada punto en el gráfico principal pasando el ratón por encima de ellos. 5. El usuario puede interactuar con las estadísticas.
Postcondición	El usuario ha podido manejar con libertad los elementos mostrados.
Excepciones	Sin excepciones
Importancia	Alta

Tabla B.5: Caso de uso 5: Controlar visualizaciones.

CU-6	Registrar
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-2, RF-2.1
Descripción	Creación de una cuenta por parte de un usuario anónimo.
Precondición	Usuario con rol de anónimo (sin cuenta/sin inicio de sesión).
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en el enlace de registro de la barra de navegación. 2. Se redirige al usuario a la página de registro. 3. Se mostrará un formulario de registro. 4. El usuario introduce su nombre. 5. El usuario introduce su correo electrónico (<i>email</i>). 6. El usuario introduce una contraseña. 7. El usuario introduce la contraseña del paso anterior repetida (confirmación). 8. El usuario hace clic en el botón de envío. 9. Se redirige al usuario a la página principal con sesión iniciada.
Postcondición	El usuario ha sido registrado en el sistema (base de datos) y se encuentra con sesión iniciada.
Excepciones	<ul style="list-style-type: none"> ■ Un campo del formulario no se ha rellenado. Se impedirá enviar el formulario instando al usuario a completar todos los campos. ■ El <i>email</i> ya tiene una cuenta registrada. Se mostrará un mensaje de error debajo del campo. ■ La contraseña de confirmación no coincide. Se mostrará un mensaje de error debajo del campo.
Importancia	Media

Tabla B.6: Caso de uso 6: Registrar.

CU-7	Iniciar de sesión
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-2, RF-2.2
Descripción	Inicio de sesión en la aplicación.
Precondición	Usuario con rol de anónimo.
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en el enlace de inicio de sesión de la barra de navegación. 2. Se redirige al usuario a la página de inicio de sesión. 3. Se mostrará un formulario de inicio de sesión. 4. El usuario introduce su correo electrónico (<i>email</i>). 5. El usuario introduce su contraseña. 6. El usuario hace clic en el botón de envío. 7. Se redirige al usuario a la página principal.
Postcondición	El usuario ha iniciado sesión y se encuentra en la página principal.
Excepciones	<ul style="list-style-type: none"> ■ Un campo no se ha rellenado. Se impedirá el envío del formulario instando al usuario a completar todos los campos. ■ El <i>email</i> no está en el sistema. Se mostrará un mensaje de error debajo del campo. ■ La contraseña no coincide con la almacenada. Se mostrará un mensaje de error debajo del campo.
Importancia	Media

Tabla B.7: Caso de uso 7: Iniciar de sesión.

CU-8	Cerrar sesión
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-2, RF-2.3
Descripción	Cerrar sesión en la aplicación
Precondición	Usuario con sesión iniciada.
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en su nombre usuario en la barra de navegación. 2. El usuario hace clic en «Cerrar sesión» en el desplegable mostrado. 3. Se redirige al usuario a la página principal y ya no tiene sesión iniciada.
Postcondición	El usuario ya no tiene sesión y se encuentra en la página principal.
Excepciones	Sin excepciones
Importancia	Media

Tabla B.8: Caso de uso 8: Iniciar de sesión.

CU-9	Personalizar perfil
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-3
Descripción	Edición de los datos personales del usuario.
Precondición	Usuario con sesión iniciada.
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en su nombre usuario en la barra de navegación. 2. El usuario hace clic en «Perfil» en el desplegable mostrado. 3. Se redirige al usuario a la página de su perfil. 4. Se mostrará un formulario de edición. 5. El usuario modifica su nombre, <i>email</i> y contraseña (cada uno de manera opcional). 6. El usuario introduce su contraseña actual. 7. El usuario hace clic en el botón de envío.
Postcondición	Los datos del usuario se han actualizado en la base de datos y mantiene su sesión.
Excepciones	<ul style="list-style-type: none"> ■ No se ha introducido contraseña actual. En este caso, se impedirá el envío del formulario instando al usuario a rellenar ese campo. ■ El <i>email</i> ha sido modificado y el nuevo ya está en el sistema. En este caso, se mostrará un mensaje de error debajo del campo (no conllevará cambios). ■ La contraseña actual no coincide con la almacenada. En este caso, se mostrará un mensaje de error debajo del campo y no se modificarán los datos.
Importancia	Baja

Tabla B.9: Caso de uso 9: Personalizar perfil.

CU-10	Acceder espacio personal
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-4, RF-4.1, RF-4.2
Descripción	Acceso a al espacio personal del usuario donde visualizar sus conjuntos de datos y ejecuciones realizadas anteriormente.
Precondición	Usuario con sesión iniciada.
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en su nombre usuario en la barra de navegación. 2. El usuario hace clic en «Mi Espacio» en el desplegable mostrado. 3. Se redirige al usuario a la página del espacio personal. 4. Se mostrarán las tablas de sus conjuntos de datos y sus ejecuciones realizadas.
Postcondición	El usuario puede visualizar sus conjuntos de datos subidos y sus ejecuciones.
Excepciones	Los datos no pueden recuperarse de la base de datos, se mostrará el error para después volver a la página principal
Importancia	Media

Tabla B.10: Caso de uso 10: Acceder espacio personal.

CU-11	Ejecutar conjunto de datos
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-4.1.1
Descripción	Utilizar un conjunto de datos subido para ejecutar uno de los algoritmos.
Precondición	Encontrarse en el espacio personal (caso de uso 10).
Acciones	<ol style="list-style-type: none"> 1. El usuario decide qué conjunto de datos utilizar. 2. El usuario pulsa en el botón con el símbolo <i>play</i>. 3. Se mostrará un «modal» con los algoritmos. 4. El usuario hace clic en uno de los algoritmos. 5. Se redirige al usuario a la página de configuración del algoritmo. 6. Ejecutar caso de uso 1 desde el paso 3.
Postcondición	El usuario ha sido redirigido a la configuración del algoritmo seleccionado. El sistema muestra además el fichero seleccionado.
Excepciones	Sin excepciones (ver excepciones del caso de uso 1)
Importancia	Media

Tabla B.11: Caso de uso 11: Ejecutar conjunto de datos.

CU-12	Replicar ejecución
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-4.2.1
Descripción	Ejecutar de nuevo una ejecución previa exactamente como ocurrió.
Precondición	Encontrarse en el espacio personal (caso de uso 10).
Acciones	<ol style="list-style-type: none"> 1. El usuario decide qué ejecución replicar. 2. El usuario pulsa en el botón con el símbolo de <i>refresh</i>. 3. Se redirige al usuario directamente a la página de visualización del algoritmo. 4. Ejecutar el caso de uso 1 desde el paso 4.
Postcondición	El usuario ha sido redirigido a la visualización de la ejecución (del algoritmo).
Excepciones	Sin excepciones (ver excepciones del caso de uso 1)
Importancia	Media

Tabla B.12: Caso de uso 12: Replicar ejecución.

CU-13	Acceder panel de administración
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-5, RF-5.1, RF-5.1.3, RF-5.2, RF-5.2.2, RF-5.3, RF-5.3.2
Descripción	Acceso a al panel de administración del usuario con rol de administrador donde visualizar todos los usuarios, los conjuntos de datos y ejecuciones realizadas.
Precondición	Usuario con sesión iniciada con rol de administrador.
Acciones	<ol style="list-style-type: none"> 1. El administrador hace clic en su nombre usuario en la barra de navegación. 2. El administrador hace clic en «Panel de administración» en el desplegable mostrado. 3. Se redirige al administrador a la página del panel de administración (<i>dashboard</i>). 4. Se mostrarán las tablas de los usuarios, conjuntos de datos y ejecuciones realizadas (de todos los usuarios). 5. Se mostrarán estadísticas básicas: <ul style="list-style-type: none"> ■ Número total de usuarios. ■ Ficheros (conjuntos de datos) subidos en los últimos 7 días. ■ Ejecuciones realizadas en los últimos 7 días.
Postcondición	El administrador tiene una vista global de todas las tablas con sus estadísticas básicas.
Excepciones	Los datos no pueden recuperarse de la base de datos, se mostrará el error para después volver a la página principal
Importancia	Media

Tabla B.13: Caso de uso 13: Acceder panel de administración.

CU-14	Eliminar conjunto de datos
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-4.1.2, RF-5.1.2
Descripción	Eliminar un conjunto de datos (fichero) del sistema.
Precondición	Encontrarse en el espacio personal (caso de uso 10) o, si tiene rol de administrador, en el panel de administración (caso de uso 13).
Acciones	<ol style="list-style-type: none"> 1. El usuario decide qué conjunto de datos eliminar sobre la tabla de conjuntos de datos. 2. El usuario pulsa en el botón rojo con el símbolo de papelera en esa fila. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario confirma la operación. 5. El fichero se ha eliminado y ha desaparecido de la tabla.
Acciones alternativas	<ol style="list-style-type: none"> 1. El usuario decide qué conjunto de datos eliminar. 2. El usuario pulsa en el botón rojo con el símbolo de <i>papelera</i>. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario cancela la operación. 5. Todo se encuentra en el mismo estado que al iniciar el caso de uso.
Postcondición	Si el usuario ha decidido eliminar el fichero, la fila ha sido eliminada y el fichero ha sido borrado del sistema junto con su referencia en la base de datos. En caso contrario, todo se encuentra como al principio.
Excepciones	No ha sido posible eliminar el fichero (sistema y/o base de datos). En este caso se muestra un mensaje de error para informar al usuario.
Importancia	Media

Tabla B.14: Caso de uso 14: Eliminar conjunto de datos.

CU-15	Eliminar ejecución
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-4.2.2, RF-5.2.2
Descripción	Eliminar una ejecución previa del sistema.
Precondición	Encontrarse en el espacio personal (caso de uso 10) o, si tiene rol de administrador, en el panel de administración (caso de uso 13).
Acciones	<ol style="list-style-type: none"> 1. El usuario decide qué ejecución eliminar sobre la tabla de ejecuciones. 2. El usuario pulsa en el botón rojo con el símbolo de papelera en esa fila. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario confirma la operación. 5. La ejecución se ha eliminado y ha desaparecido de la tabla.
Acciones alternativas	<ol style="list-style-type: none"> 1. El usuario decide qué ejecución eliminar sobre la tabla de ejecuciones. 2. El usuario pulsa en el botón rojo con el símbolo de <i>papelera</i> en esa fila. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario cancela la operación. 5. Todo se encuentra en el mismo estado que al iniciar el caso de uso.
Postcondición	Si el usuario ha decidido eliminar la ejecución, la fila ha sido eliminada y los datos de la ejecución han sido borrados del sistema junto con su referencia en la base de datos. En caso contrario, todo se encuentra como al principio.
Excepciones	No ha sido posible eliminar la ejecución (sistema y/o base de datos). En este caso se muestra un mensaje de error para informar al usuario.
Importancia	Media

Tabla B.15: Caso de uso 15: Eliminar ejecución.

CU-16	Editar perfil ajeno
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-5.1.1
Descripción	Edición de los datos personales de cualquier usuario del sistema.
Precondición	Encontrarse en el panel de administración (caso de uso 13).
Acciones	<ol style="list-style-type: none"> 1. El administrador decide qué usuario editar. 2. El administrador pulsa en el botón verde con el símbolo de lápiz en esa fila. 3. Se redirige a la página de personalización del perfil. 4. Se muestra al administrador el punto de vista del usuario que se está editando con un indicativo. 5. El administrador modifica los campos que desee del usuario. 6. El administrador envía el formulario. 7. Se actualizan los datos del usuario. 8. Se redirige al administrador al panel de administrador.
Postcondición	Los datos del usuario se han actualizado en la base de datos y el administrador se encuentra en el panel de administrador.
Excepciones	<ul style="list-style-type: none"> ■ El email ha sido modificado y el nuevo ya está en el sistema. En este caso, se mostrará un mensaje de error debajo del campo.
Importancia	Baja

Tabla B.16: Caso de uso 16: Editar perfil ajeno.

CU-17	Eliminar usuario
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-5.1.2
Descripción	Eliminar un usuario del sistema.
Precondición	Encontrarse en el panel de administración (caso de uso 13).
Acciones	<ol style="list-style-type: none"> 1. El usuario decide qué usuario eliminar sobre la tabla de usuarios. 2. El usuario pulsa en el botón rojo con el símbolo de papelera en esa fila. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario confirma la operación. 5. La ejecución se ha eliminado y ha desaparecido de la tabla.
Acciones alternativas	<ol style="list-style-type: none"> 1. El usuario decide qué usuario eliminar sobre la tabla de ejecuciones. 2. El usuario pulsa en el botón rojo con el símbolo de <i>papelera</i> en esa fila. 3. Se mostrará un «modal» para confirmar la operación. 4. El usuario cancela la operación. 5. Todo se encuentra en el mismo estado que al iniciar el caso de uso.
Postcondición	Si el usuario ha decidido eliminar el usuario, la fila ha sido eliminada y los datos del usuario (incluidos ficheros y ejecuciones) han sido borrados del sistema junto con su referencia en la base de datos. En caso contrario, todo se encuentra como al principio.
Excepciones	No ha sido posible eliminar el usuario (sistema y/o base de datos). En este caso se muestra un mensaje de error para informar al administrador.
Importancia	Media

Tabla B.17: Caso de uso 17: Eliminar usuario.

CU-18	Cambiar idioma
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-6
Descripción	Cambiar el idioma de la página (entre inglés y español)
Precondición	No hay precondiciones
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en el símbolo de traducción en la barra de navegación. 2. Aparece un desplegable de idiomas. 3. El usuario hace clic en el idioma deseado. 4. Se actualiza la página indicando el idioma a la aplicación.
Postcondición	La página se ha cambiado al idioma seleccionado
Excepciones	Sin excepciones
Importancia	Baja

Tabla B.18: Caso de uso 18: Cambiar idioma.

CU-19	Consultar ayuda
Versión	1.0
Autor	David Martínez Acha
Requisitos asociados	RF-7
Descripción	Acceder al manual de usuario de la aplicación
Precondición	No hay precondiciones
Acciones	<ol style="list-style-type: none"> 1. El usuario hace clic en el símbolo de la interrogación en la barra de navegación. 2. El usuario es redireccionado (en una nueva pestaña) al manual de usuario (PDF).
Postcondición	El usuario visualiza el manual de usuario
Excepciones	Sin excepciones
Importancia	Baja

Tabla B.19: Caso de uso 19: Consultar ayuda.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se van a describir las decisiones de diseño tomadas para llevar a cabo todos los objetivos y requisitos iniciales establecidos. Se presentará el formato que se utiliza para tratar los datos, cuál es el procedimiento interno que realiza la Web para ofrecer al usuario las visualizaciones y cómo se traduce todo ello en la arquitectura subyacente.

C.2. Diseño de datos

Información de entrenamiento de los algoritmos

Dado que el proyecto maneja dos fuentes de datos claras (por un lado los algoritmos y por otro la base de datos), esta sección se ha separado del mismo modo para explicar cada una de las partes: la información de entrenamiento y las entidades utilizadas en base de datos.

Todas las visualizaciones de los algoritmos se nutren del proceso de entrenamiento. Es decir, se tuvo que diseñar una forma de aglutinar la información que ocurre durante este proceso.

Para adelantar un poco el funcionamiento de la web, todos estos datos son transformados a JSON, el formato de texto con la sintaxis de JavaScript que resulta muy sencillo para el intercambio de datos. Además, a la hora de trabajar con ello en dicho lenguaje, se hace de forma directa (como diccionarios en otros lenguajes de programación).

Pero antes de esa transformación, todos los datos son generados en Python. La estructura de datos por excelencia para almacenar muchos datos (gracias a su multitud de opciones) son los «DataFrames». Toda la información que generan los algoritmos son de este tipo.

Cada algoritmo retorna varios de estos «DataFrame», al menos uno del proceso de etiquetado y otro con las estadísticas generales. En el caso de *Democratic Co-Learning* y *Tri-Training*, estos además retornan los de las estadísticas específicas de cada clasificador base.

Etiquetas

Este «DataFrame» contiene todas las operaciones en las que los clasificadores han añadido nuevas etiquetas a instancias no etiquetadas. Por lo general indicarán los momentos en los que fueron etiquetadas (iteración) y el valor de las mismas.

Self-Training Este formato sirve como base para todos los siguientes. Es una idea muy sencilla (pues *Self-Training* también lo es).

La estructura de datos contendrá múltiples filas, cada una representa cada instancia de todo el conjunto de datos de entrenamiento. Lo que se entiende por conjunto de entrenamiento es, todos los datos etiquetados de los que aprenderá el clasificador base junto con los no etiquetados.

Ahora bien, la información útil que permitirá saber los momentos de clasificación y/o etiquetas estarán en las columnas:

- Columnas con los nombres de los atributos de las instancias. Por ejemplo, para el famoso conjunto de datos *Iris* se tendrán 4 columnas (una por cada atributo, **no se incluye la clase**): «*sepal length*», «*sepal width*», «*petal length*» y «*petal width*».
- Columna «*iter*»: Por cada fila, representa el número de la iteración en la que esa instancia fue clasificada. Si por el criterio de parada no llega a ser clasificada corresponderá con el número de iteraciones final + 1. Es decir, si en la iteración 9 el entrenamiento finalizó, corresponderá con 10. Obviamente, si la instancia es un dato inicial tendrá como iteración 0.
- Columna «*target*»: Representa la etiqueta de la instancia. Es importante destacar que esta siempre será de tipo entero. Esto es porque en

pasos previos se eliminan los valores nominales (no están permitidos). Si tampoco llega a ser clasificado, corresponderá con «-1».

En la tabla C.1 se muestra un ejemplo del formato para Self-Training (entrenamiento finalizado en la iteración 9).

	petal.length	petal.width	sepal.length	sepal.width	iter	target
0	5.6	2.5	3.9	1.1	0	1
1	6.1	2.8	4.0	1.3	0	2
78	6.7	3.1	5.6	2.4	6	2
79	6.8	3.2	5.7	2.5	7	1
142	6.8	2.8	4.8	1.4	10	-1

Tabla C.1: Ejemplo de DataFrame de Self-Training

En la tabla C.1 se puede ver un extracto de lo que podría ser una ejecución. Las columnas de los atributos, la iteración (con iteración 10 para denotar que no fue etiquetado) y la etiqueta o *target* (con -1 para los que no fueron etiquetados).

Co-Training Es muy similar a *Self-Training* salvo que este algoritmo concreto utiliza dos clasificadores base. La consecuencia de esto a nivel del formato es que se necesita almacenar cuál de los dos clasificadores clasifica cada instancia. Pero más allá de esto, es el mismo formato.

Recopilando, las columnas para el «DataFrame» de *Co-Training* son:

- Columnas con los nombres de los atributos de las instancias.
- Columna `iter`.
- Columna `target`.
- Columna `clf`: Esta nueva columna indica el clasificador que le dio el valor a la etiqueta. Por convención (necesaria para otros procesos en Web) es que si el dato es inicial, `clf` valdrá «inicio», si es un dato clasificado durante el proceso valdrá «CLF(*nombre_clasificador*)» donde *nombre_clasificador* será el extraído de *Scikit-Learn*. Si no llega a ser clasificado valdrá -1.

En la tabla C.2 se muestra un ejemplo del formato para Co-Training (entrenamiento finalizado en la iteración 9).

	petal.length	petal.width	sepal.length	sepal.width	iter	target	clf
0	5.6	2.5	3.9	1.1	0	1	inicio
1	6.1	2.8	4.0	1.3	0	2	inicio
78	6.7	3.1	5.6	2.4	6	2	CLF(SVC)
79	6.8	3.2	5.7	2.5	7	1	CLF(GaussianNB)
142	6.8	2.8	4.8	1.4	10	-1	-1

Tabla C.2: Ejemplo de DataFrame de Co-Training

Democratic Co-Learning Tomando como base Co-Training, este algoritmo no añade ninguna columna más (ni elimina), solo modifica las existentes. Concretamente, las tres últimas columnas («iter», «target» y «clf») que estaban en singular ahora pasan a plural: «iters», «targets» y «clfs».

La razón es simplemente para mantener una lógica y semántica interna, este algoritmo «comparte» las instancias entre tres clasificadores base. Cada uno de ellos puede clasificar cada instancia, incluso con distintas etiquetas y en la misma iteración que los otros clasificadores. Por lo tanto, para cada una de estas columnas y para cada instancia se tendrá ahora una lista.

Recopilando, las columnas para el «DataFrame» de Democratic Co-Learning son:

- Columnas con los nombres de los atributos de las instancias.
- Columna **iters**: Lista con la iteración en la que cada clasificador etiqueta la instancia. En el caso de un dato inicial, esta lista solo tendrá una posición y contendrá 0 ([0]), para el resto siempre tendrá tres posiciones (por los tres clasificadores base). Para este último caso cada posición es independiente, es decir, si un clasificador base no ha etiquetado, contendrá -1 (**diferencia con los anteriores**¹), pero para esa misma instancia otro clasificador base sí que puede haber etiquetado y contendrá dicha iteración (por ejemplo: [-1, 4, -1])
- Columna **targets**: Exactamente igual a «iters» salvo que no indica la iteración, sino la etiqueta que asigna el clasificador base. De nuevo, si no asigna una, su posición contendrá -1.

¹La razón de que se indique -1 en vez de número de iteraciones + 1, es porque toda esta estructura se genera antes del entrenamiento. En los algoritmos anteriores, si quedaba alguna sin etiquetar, se añadía al final y se sabía el número de iteraciones final.

- Columna `clfs`: Hasta ahora se ha hablado de posiciones en las dos columnas anteriores. Para saber a qué clasificador se refieren esas posiciones, esta columna contiene otra lista con los tres nombres de los clasificadores. Si es un dato inicial, contendrá [inicio], en otro caso contendrá algo como [CLF1(KNeighborsClassifier), CLF2(DecisionTreeClassifier), CLF3(GaussianNB)].

En la tabla C.3 se muestra un ejemplo del formato para Democratic Co-Learning (entrenamiento finalizado en la iteración 9)²:

	petal.length	petal.width	sepal.length	sepal.width	iter	target	clf
0	5.6	2.5	3.9	1.1	[0]	[1]	[inicio]
1	6.1	2.8	4.0	1.3	[0]	[2]	[inicio]
78	6.7	3.1	5.6	2.4	[1, 4, -1]	[2, 2, -1]	[CLF1, CLF2, CLF3]
79	6.8	3.2	5.7	2.5	[-1, 2, -1]	[-1, 1, -1]	[CLF1, CLF2, CLF3]
142	6.8	2.8	4.8	1.4	[-1, -1, -1]	[-1, -1, -1]	[CLF1, CLF2, CLF3]

Tabla C.3: Ejemplo de DataFrame de Democratic Co-Learning

Tri-Training Tomando la idea de Democratic Co-Learning, en este algoritmo también se tienen tres clasificadores que pueden clasificar individualmente cada instancia. Sin embargo, cada iteración, el conjunto de datos **nuevos** etiquetados se vacía. Es decir, en una iteración se acaban etiquetando algunos, pero al principio de la siguiente se vacía ese conjunto y se vuelven a etiquetar de nuevo.

Entonces, el mecanismo de las listas de Democratic Co-Learning no funcionaría. Se debe añadir un nivel más de registro. Simplemente con añadir una lista en cada posición de las listas de «iters» y «targets» ya se puede registrar todos los momentos en los que una instancia se etiqueta (de nuevo, cada instancia podría ser etiquetada dos o más veces por un mismo clasificador base, al contrario del anterior).

Recopilando, las columnas para el «DataFrame» de Tri-Training son:

- Columnas con los nombres de los atributos de las instancias.
- Columna «iters»: Lista de listas con la iteración en la que cada clasificador etiqueta la instancia. En el caso de un dato inicial, esta lista solo tendrá una posición y contendrá 0 ([0]), para el resto siempre tendrá una lista en las tres posiciones (por los tres clasificadores base). Para

²Se han acordado los nombres de los clasificadores [CLF1, CLF2, CLF3], debería ser [CLF1(KNeighborsClassifier), CLF2(DecisionTreeClassifier), CLF3(GaussianNB)]

esto último caso las posiciones siguen siendo independientes solo que ahora si no se etiqueta la lista estará vacía (por ejemplo, las iteraciones de un dato podrían ser: `[[2], [], []]`, los dos últimos clasificadores no etiquetaron esa instancia en ningún momento). Para clarificar lo comentado anteriormente, los clasificadores podrían etiquetar una misma instancia dos o más veces (por ejemplo: `[[2], [2], [1, 2]]`, el último clasificador etiquetó la instancia en dos ocasiones).

- Columna «targets»: Exactamente igual a «iters» salvo que no indica la iteración, sino la etiqueta que asigna el clasificador base. Y de nuevo, si un clasificador no etiqueta en ningún momento esa instancia su lista interna estará vacía (por ejemplo: `[[2], [], [2]]`).
- Columna «clfs»: No se modifica respecto a Co-Training, contiene la lista de los nombres de los tres clasificadores.

En la tabla C.4 se muestra un ejemplo del formato para Tri-Training (entrenamiento finalizado en la iteración 9)³:

	petal.length	petal.width	sepal.length	sepal.width	iter	target	clf
0	5.6	2.5	3.9	1.1	[0]	[1]	[inicio]
1	6.1	2.8	4.0	1.3	[0]	[2]	[inicio]
78	6.7	3.1	5.6	2.4	[[1], [1,2], []]	[[2], [2, 2], []]	[CLF1, CLF2, CLF3]
79	6.8	3.2	5.7	2.5	[[], [2], []]	[[], [1], []]	[CLF1, CLF2, CLF3]
142	6.8	2.8	4.8	1.4	[[], [], []]	[[], [], []]	[CLF1, CLF2, CLF3]

Tabla C.4: Ejemplo de DataFrame de Tri-Training

Interpretación sencilla de estos formatos Cuando se realiza su visualización, el primer paso es extraer los datos. Lo que se hace es crear un punto en el gráfico por cada instancia, con la particularidad de que cuando se entra en los algoritmos Democratic Co-Learning o Tri-Training, se añade más de un punto por cada instancia, representando la individualidad de cada clasificador base (cada uno de ellos puede haber clasificado esa instancia por separado e incluso más de una vez).

Cuando se genera el gráfico con sus puntos, cada punto lleva guardada la información de las iteraciones, etiquetas y clasificadores. Así se controla cuándo ocultar/colorear/mostrar un punto. Por ejemplo, cuando se navega a la siguiente iteración se filtran todos los puntos que tenga esa iteración en la columna «iter» (o «iters»). Obviamente, cada algoritmo tiene sus particularidades, pero esta es la idea general.

³Se han acertado los nombres de los clasificadores [CLF1, CLF2, CLF3] debería ser [CLF1(KNeighborsClassifier), CLF2(DecisionTreeClassifier), CLF3(GaussianNB)]

	Accuracy	Precision	Error	F1_score	Recall
0	0.833333	0.888889	0.166667	0.822222	0.833333
1	1.000000	1.000000	0.000000	1.000000	1.000000
2	1.000000	1.000000	0.000000	1.000000	1.000000
3	1.000000	1.000000	0.000000	1.000000	1.000000
4	1.000000	1.000000	0.000000	1.000000	1.000000

Tabla C.5: Ejemplo de DataFrame de las estadísticas generales

Estadísticas generales

Las estadísticas generales son comunes a todos los algoritmos, no hay ninguna modificación entre ellos.

Se tiene un «DataFrame» con tantas filas como iteraciones se han ejecutado. En cuanto a las columnas, simplemente son los nombres de las estadísticas que se desean mostrar. Los nombres son los que se mostrarán en la Web.

En la tabla C.5 se presenta un ejemplo del formato de las estadísticas generales.

Estadísticas específicas

Este tipo de estadísticas se refiere a la necesidad de mostrar también las estadísticas particulares de los clasificadores base. Esto aplica para Co-Training, Democratic Co-Learning y Tri-Training.

El núcleo de esta estructura de datos siguen siendo los «DataFrames», sin embargo, se han envuelto en un diccionario. El diccionario tiene como claves cada uno de los nombres de los clasificadores base de la ejecución (por ejemplo: CLF3(GaussianNB))⁴. Los valores del diccionario serán esos «DataFrames» que son exactamente iguales que para las estadísticas generales. Guardarán por cada iteración (fila), las estadísticas (columnas) para el clasificador base concreto (el de la clave).

Base de datos

El diagrama general E/R se puede ver en la figura C.1. Para el manejo de toda la información que se ha necesitado en el flujo de la aplicación, se han creado las siguientes entidades:

⁴En Co-Training nunca podría llegar a ser «CLF3» pues solo hay dos clasificadores, pero es la misma idea

- **Usuario** (User): Representa al usuario registrado en la aplicación. Contiene un identificador que representa la clave primaria. Los datos personales de los usuarios que guarda la entidad son: email (único en toda la base de datos), el nombre y la contraseña (cifrada mediante SHA-256). Aparte de estos, también se guarda el último inicio de sesión y un campo que indica si el usuario es administrador.

Por último, un usuario puede tener varios ficheros y varias ejecuciones. Esto está ejemplificado mediante dos relaciones a las otras dos entidades siguientes (relaciones **One-To-Many**).

- **Dataset**: Representa un fichero subido por el usuario. Contiene un identificador que representa la clave primaria. Como datos propios del fichero, contiene el nombre del mismo (tal y como se guarda en el sistema) y la fecha de subida.

Esta entidad mantiene una clave foránea al usuario (el identificador del usuario) junto con una referencia directa a él (aspecto interno de *SQLAlchemy* para poder acceder al usuario sin realizar consulta).

- **Run**: Representa la ejecución de un algoritmo. Contiene un identificador que representa la clave primaria. Como datos propios de una ejecución contiene:

- El nombre del algoritmo.
- La cadena de caracteres (JSON) con todos los parámetros.
- El nombre del fichero con el que se ejecutó (conjunto de datos).
- La fecha de ejecución.
- Componente *X*.
- Componente *Y*.
- Nombre del fichero del sistema que guarda todos los datos de la ejecución.

Esta entidad también mantiene una clave foránea al usuario (el identificador del usuario) junto con una referencia directa a él (aspecto interno de *SQLAlchemy* para poder acceder al usuario sin realizar consulta).

Diagrama Entidad/Relación

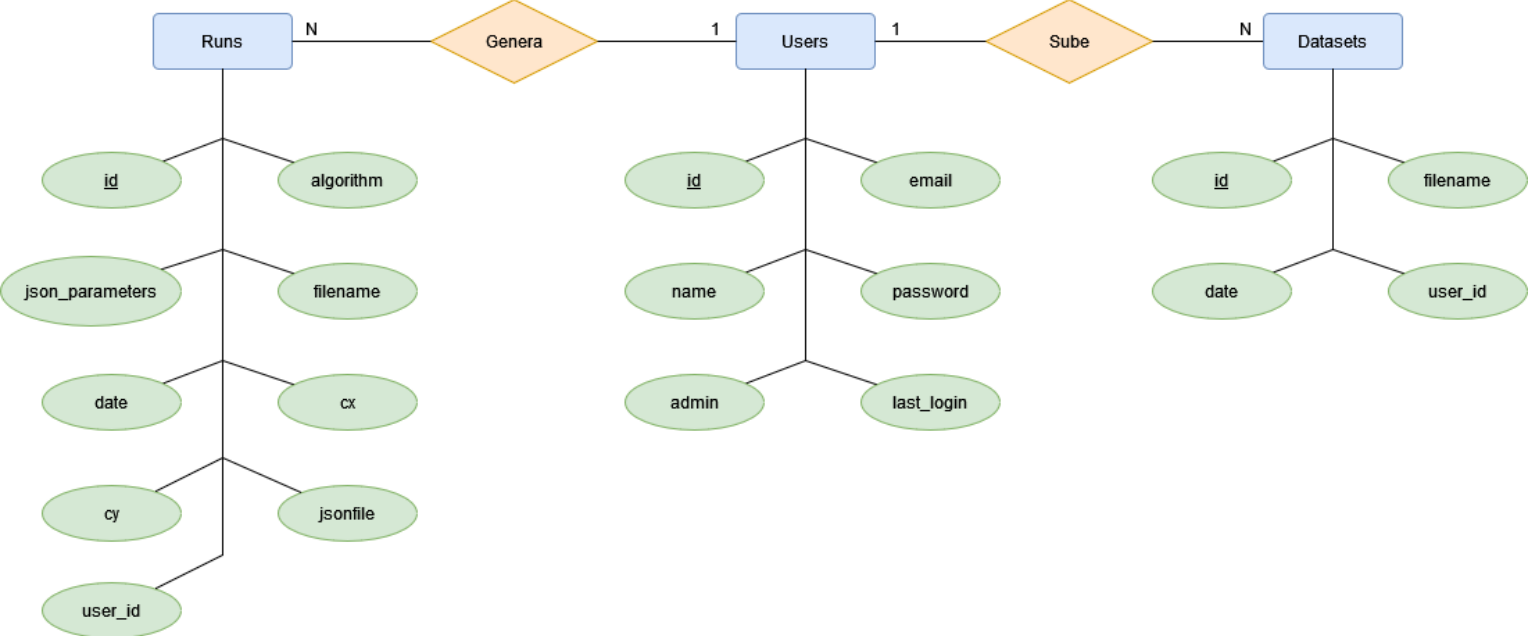


Figura C.1: Diagrama Entidad/Relación

Diagrama relacional

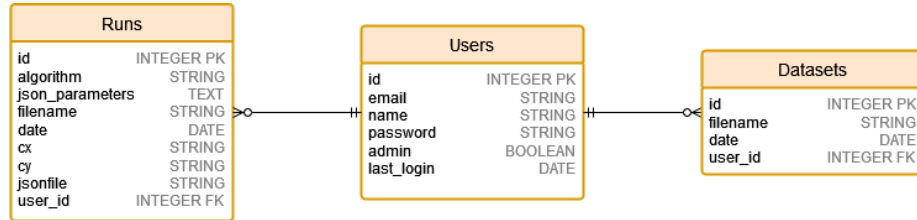


Figura C.2: Diagrama relacional

C.3. Diseño procedimental

En esta sección se incluyen los diagramas de secuencia considerados para explicar la interacción del usuario con la aplicación (el procedimiento seguido).

Concretamente, la interacción principal de la aplicación consiste en la visualización de un algoritmo. El diagrama de secuencia de dicho proceso puede verse en la figura C.3. Esta interacción puede resumirse en:

1. El usuario accede a la aplicación y selecciona un algoritmo. Por su parte, el servidor, establecerá que el usuario lo ha seleccionado.
2. El usuario carga un conjunto de datos que el servidor almacena y le permite al usuario configurar el algoritmo.
3. El usuario introduce los parámetros del algoritmo (propios del algoritmo y de la visualización).
4. El navegador realizará una petición con todos esos parámetros a la espera de obtener una respuesta. Mientras ocurre, el servidor se encarga de instanciar el algoritmo, ejecutarlo y enviar los resultados al navegador.
5. El navegador (mediante JavaScript) interpretará los datos y brindará las visualizaciones al usuario.

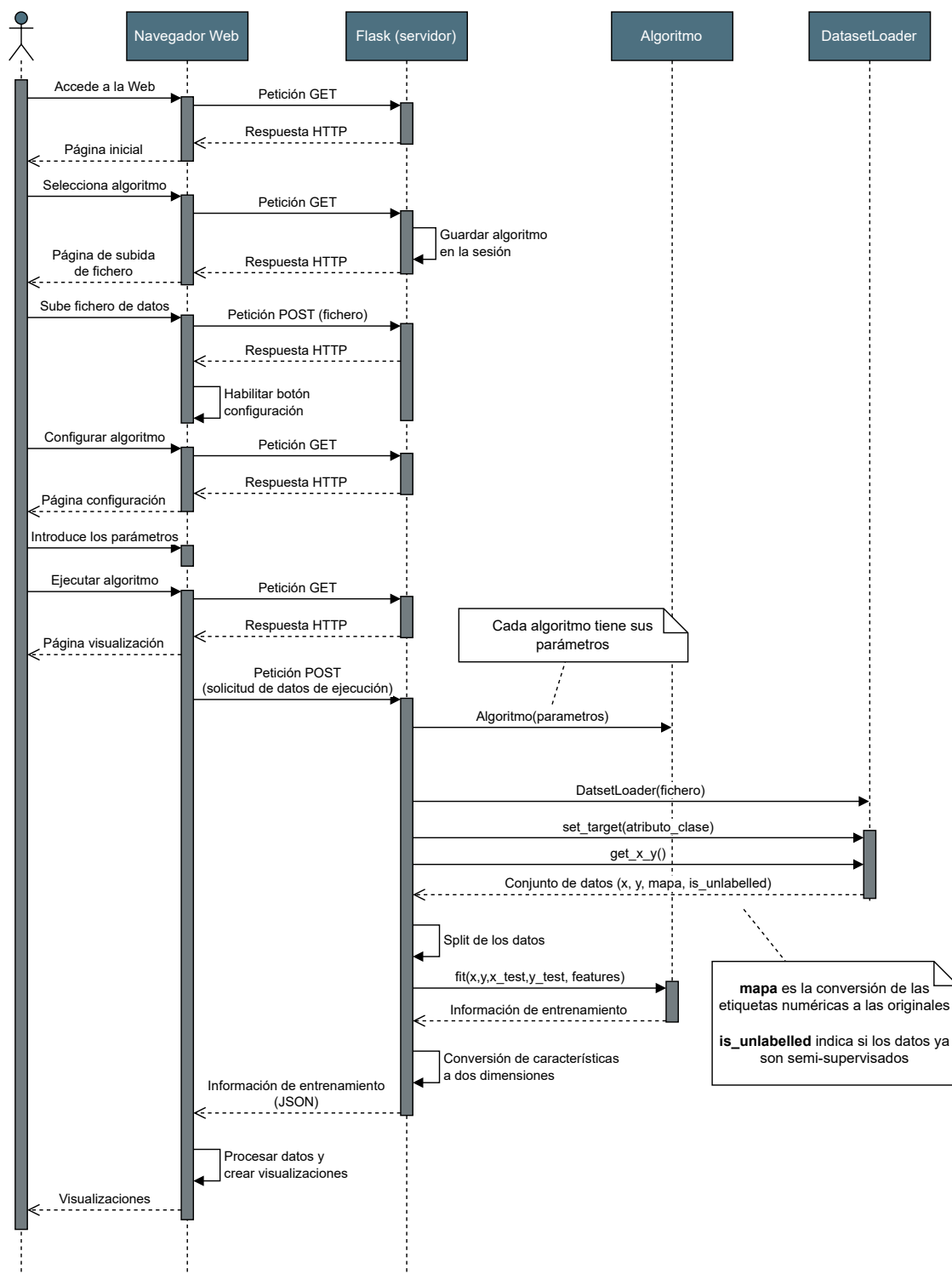


Figura C.3: Diagrama de secuencia de la visualización de un algoritmo

C.4. Diseño arquitectónico

Adelantando el contenido de este apartado, la arquitectura seguida en el desarrollo del software es la arquitectura de tres capas. Sin embargo, esta arquitectura tiene como punto de partida la arquitectura *cliente-servidor*.

La arquitectura cliente-servidor es un modelo de diseño en el que las tareas se reparten entre los servidores (proveedores) y los clientes (demandantes) [7]. De hecho, como en principio solo tienen estos dos elementos, suele referirse a veces como arquitectura de dos capas.

La separación entre ambos elementos es lógica y además el servidor no tiene por qué ser una única máquina. Y aquí es donde entran en juego los cimientos de la arquitectura de tres capas. Una disposición muy utilizada de cliente-servidor son los sistemas multicapa, en ellos, el servidor se descompone en más elementos (mayor desacoplamiento).

Arquitectura de tres capas

Estas tres capas o niveles son: capa de presentación, capa de aplicación o de negocio y la capa de datos. En la figura C.4 se representa un diagrama de este tipo de arquitectura.

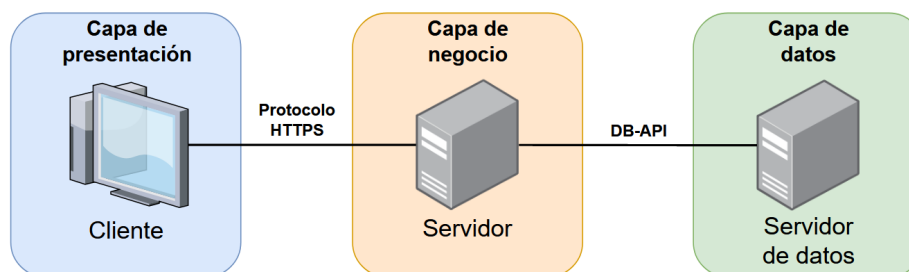


Figura C.4: Arquitectura de tres capas.

- **Capa de presentación:** es la capa con la que se presenta el sistema y el usuario interactúa. Muestra la información al usuario y además recopila información de este. Esta capa es la que ejecuta el navegador Web desde el que accede el usuario [3]. Se conoce también como interfaz gráfica (del sistema). Esta capa se comunica exclusivamente con la capa de negocio [9].

En el proyecto, esta capa está implementada mediante todas las plantillas, estilos (CSS) y código JavaScript con todos los formularios, enlaces, botones y visualizaciones. Concretamente, esto es generado

por el «framework» Flask (este «framework» puede verse en este caso como la capa de presentación). Toda esta capa se nutre de la información que solicita a la capa de negocio (por ejemplo, los datos de las ejecuciones, no son generados por la capa de presentación sino por esta última).

La comunicación entre la capa anterior y la siguiente se realiza mediante el protocolo HTTP (los conceptos pueden encontrarse en la memoria) añadiendo la segurización del mismo (HTTPS) para cifrar los mensajes. Aquí aparece el concepto de API, que define cómo se comunica el navegador con el servidor. En el proyecto actual se está utilizando una API por así decirlo, «privada», en el sentido de que no está pensada para ser accedida por todos los usuarios, sino solo los que estén manejando la aplicación y en el flujo de la misma.

- **Capa de negocio:** es el núcleo de la aplicación, se encarga de manejar las peticiones del usuario desde la capa de presentación y de enviar las respuestas correspondientes. Su nombre reside en que aquí se define la lógica empresarial (qué es lo que hace la aplicación). También se comunica con la capa de datos para almacenar o recuperar información. En el proyecto, esta capa está implementada en Python en el código de cada ruta accedida. Es decir, las peticiones que realiza el usuario serán manejadas por las rutas correspondientes, realizando el procesamiento adecuado (el negocio). Por ejemplo, la ejecución de un algoritmo dados sus parámetros para retornar los datos de la ejecución o el registro de un nuevo usuario.

La comunicación entre la capa anterior y la siguiente se realiza mediante DB-API (Python Database API). No es un protocolo como tal, sino un conjunto de clases y funciones para el acceso a múltiples bases de datos (MySQL, SQLite, PostgreSQL...). Además, no se está utilizando directamente, sino que se hace uso de «SQLAlchemy», que oculta el manejo de todas esas funciones y lo hace mucho más simple. Dependiendo del dialecto de la base de datos, cuando se intenta acceder a una base de datos, se establecerá lo que se llama una «conexión DBAPI».

- **Capa de datos:** conocida también como nivel de base de datos o nivel de acceso, almacena y gestiona el acceso a los datos. Por lo general esta capa es un sistema de gestión de base de datos como MySQL, PostgreSQL, Oracle...

En el proyecto actual se está utilizando SQLite, una biblioteca que implementa un motor de base de datos. Actúa por sí mismo como si fuera un servidor independiente [5] (simulando lo que hacen otras bases de datos), pero completamente acoplado a la aplicación (en local). Es decir, no se crea como en otras bases de datos (MySQL), un proceso o instancia separada de la aplicación (que incluso suele estar en otro servidor). La base de datos SQLite es un «todo en uno» que se encuentra en la misma estructura de la aplicación, almacenada como un archivo único. Desde el punto de vista de la aplicación (capa de negocio) esto no tiene relevancia y actúa de la misma forma que con otros gestores.

Por todo esto, el esquema de la arquitectura de tres capas podría ahora verse como en la figura C.5.

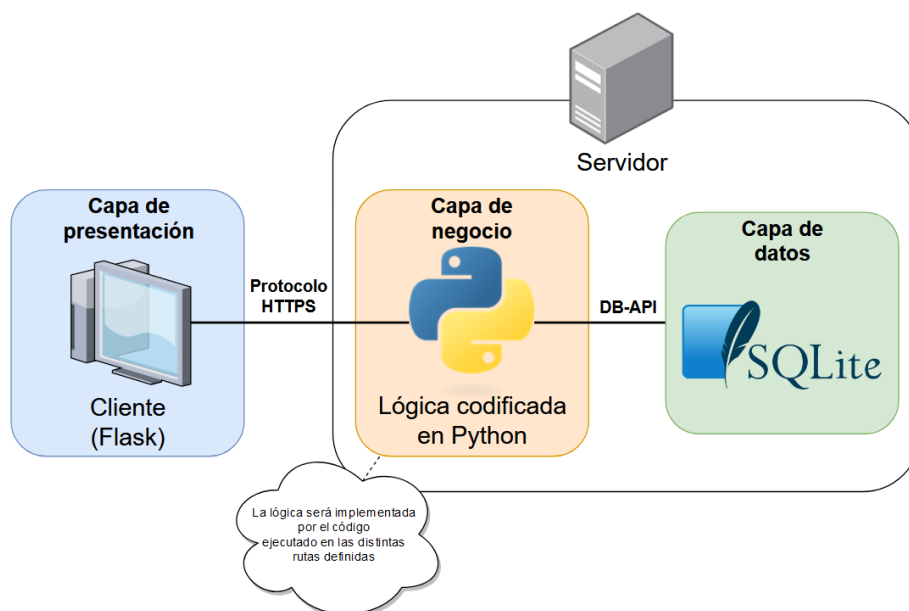


Figura C.5: Arquitectura de tres capas.

Siguen apareciendo las tres capas propias de la arquitectura, pero con la particularidad de que no se tiene un servidor dedicado para la base de datos.

Diagrama de componentes y despliegue

En el diagrama C.6 es una simplificación de los componentes que intervienen en la aplicación. Se ha abstraído la aplicación como un todo, es conveniente visualizar el diagrama de despliegue (ver figura C.7).

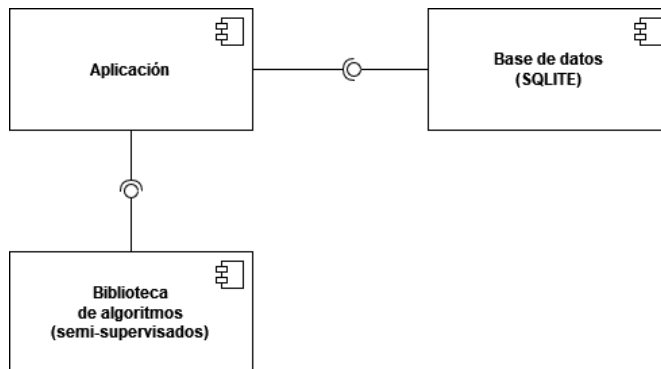


Figura C.6: Diagrama de componentes.

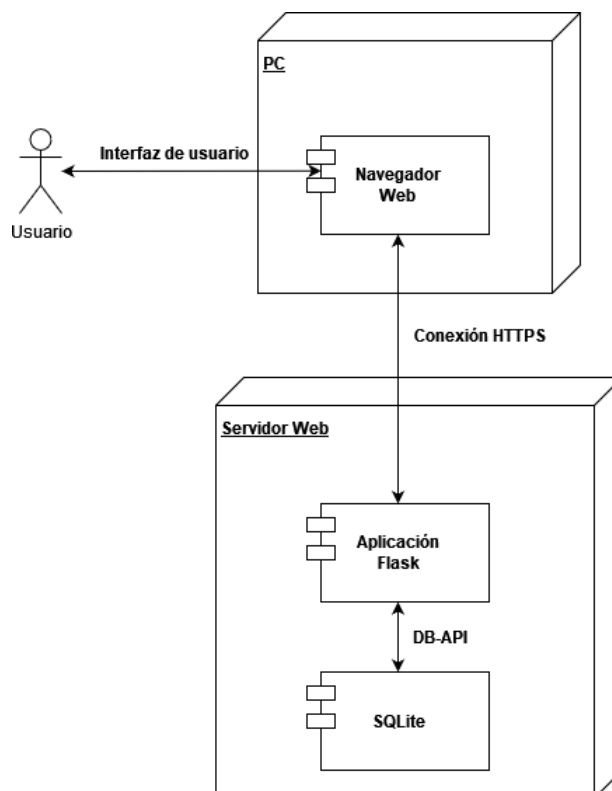


Figura C.7: Diagrama de despliegue.

C.5. Diseño Web

Primer *mockup* o maqueta

Se presenta el primer *mockup* o maqueta que se comentó de la página web. Todas las páginas tendrían una base común en la que aparecerá información general como la Universidad de Burgos (barra superior).

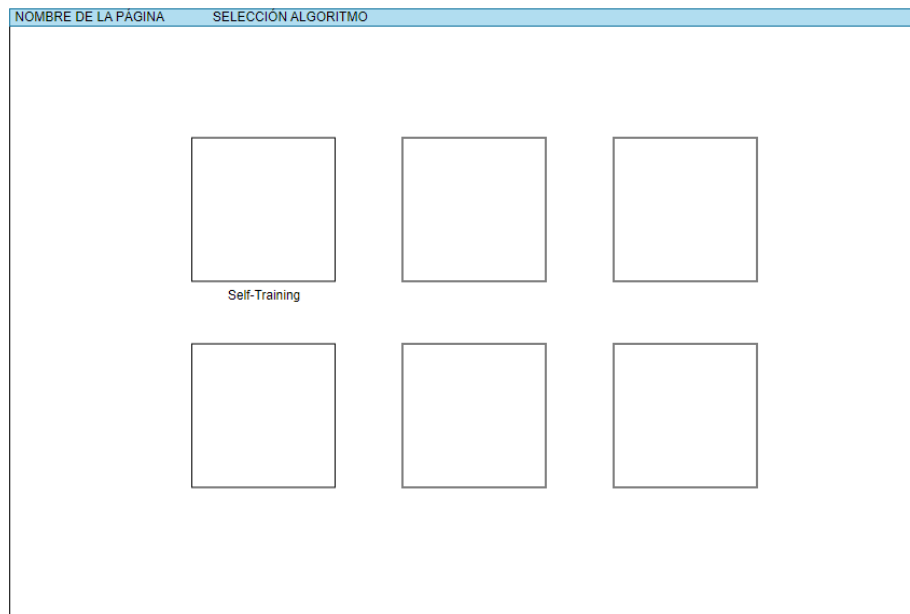



Figura C.8: Página inicial de la Web.

En esta página inicial el usuario podrá seleccionar el algoritmo que desea visualizar. En los cuadrados existirá un logo o imagen representativa del algoritmo junto con su nombre.

NOMBRE DE LA PÁGINA SELF-TRAINING

 SUBIR DATASET ARCHIVO SUBIDO

Precargado de atributos encontrados.
Usuario selecciona los parámetros

Datasets Locales ▾

Wine
Breast cancer
Otros...

Ejecutar

Explicación Self-Training

Pseudocódigo

Figura C.9: Página de configuración del algoritmo.

En esta ventana el usuario podrá subir el conjunto de datos que desee o incluso seleccionar alguno de los almacenados localmente. Además, como los algoritmos tienen parámetros personalizables también habrá elemento para configurarlos.

Antes de iniciar, se muestra una explicación del algoritmo general y su pseudocódigo.

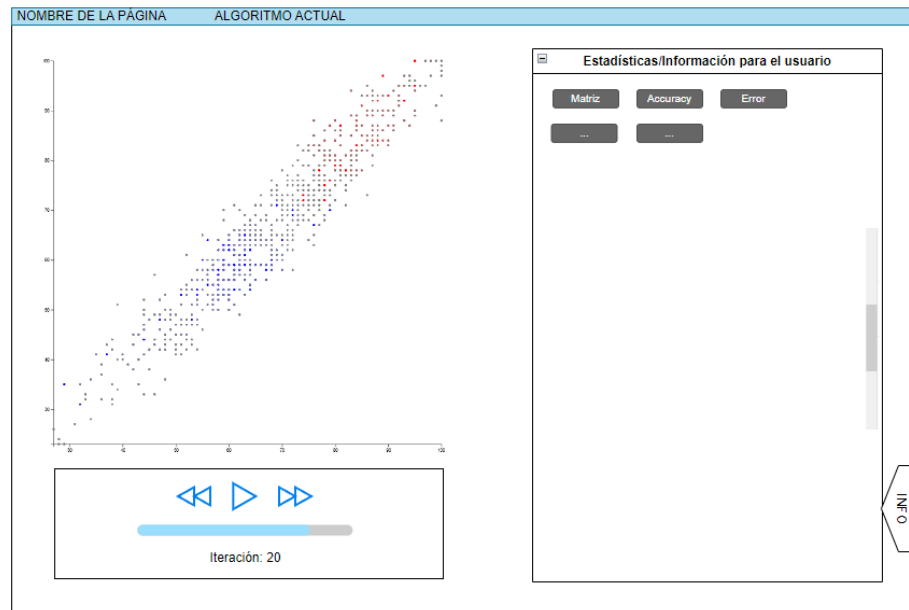


Figura C.10: Página de ejecución del algoritmo.

Mostrará la evolución del entrenamiento de los algoritmos con una vista principal (izquierda) de la clasificación y un compendio de métricas como la precisión o el error en su caso (derecha). Esto último principalmente planteado para ocultar/ver lo que el usuario desee en cada momento.

Diseño de Usuarios



Figura C.11: Barra de navegación principal



Figura C.12: Barra de navegación con usuario

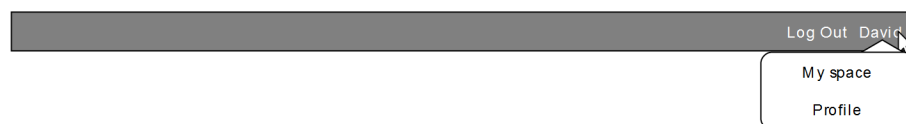


Figura C.13: Animación «hover» en barra de navegación

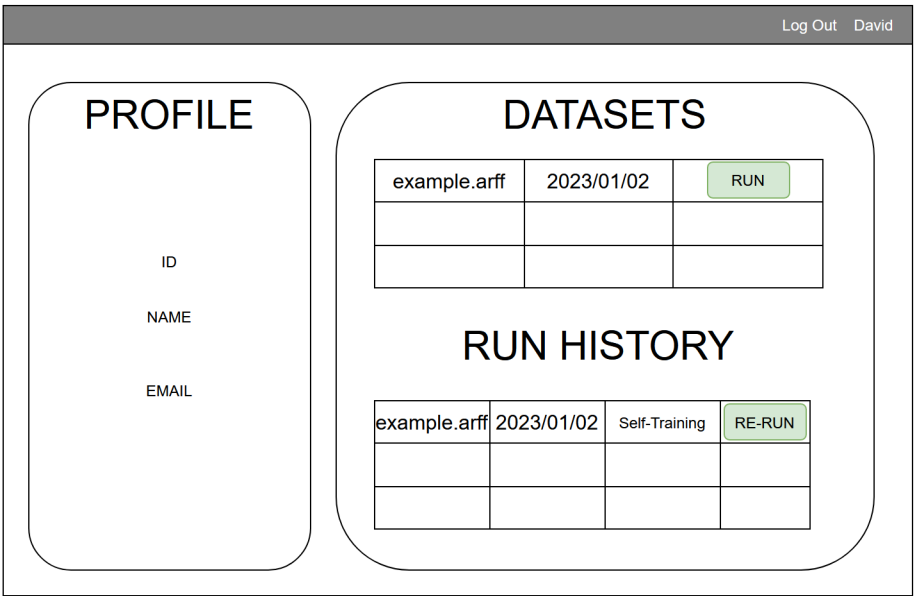


Figura C.14: Espacio personal del usuario

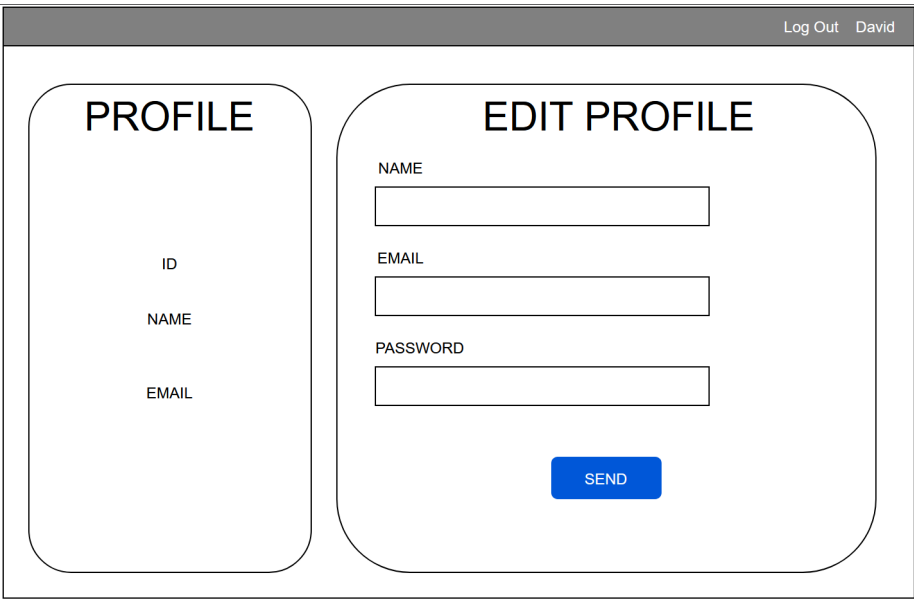


Figura C.15: visualización del perfil y modificación

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta sección se presenta toda la documentación técnica del desarrollo del proyecto. Trata de ser una guía para entender cómo se ha hecho el proyecto comenzando por los directorios y su contenido junto con un manual introductorio para un programador que es iniciado en el proyecto. También se presenta una explicación y ejemplificación de la instalación del mismo junto con las pruebas que se han realizado para su validación.

Repositorio Github

<https://github.com/dma1004/TFG-SemiSupervisado>

D.2. Estructura de directorios

Estos son los directorios en los que se organiza el proyecto:

```
/
├── algoritmos: algoritmos Semi-Supervisados
│               implementados.
│   ├── test: directorios y ficheros mediante los
│   │         que se valida el desarrollo software
│   │         correcto del proyecto.
│   │   ├── check_implementations: pruebas para la validación
│   │   │   de los algoritmos.
│   │   ├── results: ficheros CSV con resultados de
│   │   │   validación cruzada.
│   │   ├── check_utils: pruebas para la validación de las
│   │   │   utilidades.
│   │   ├── test_files: ficheros de prueba (ARFF y CSV) para
│   │   │   las pruebas.
│   │   ├── profiling: pruebas para medir el tiempo de
│   │   │   ejecución.
│   │   └── profile_results: resultados de los procesos de
│   │       «profiling».
│   └── utilidades: utilidades que realizan ciertos
│       pasos de la aplicación y de los
│       algoritmos (comunes).
├── docs: documentación teórica y técnica del
│       proyecto (hecha en  $\text{\LaTeX}$ ).
│   ├── img: imágenes utilizadas para la
│   │       generación de la documentación.
│   └── tex: archivos de texto plano con código
│        $\text{\LaTeX}$ .
├── web: aplicación web (se expandirá en las
│       siguientes páginas).
└── videos: videotutoriales de la aplicación.
```


web: aplicación web.

- **app:** contiene la definición de las rutas, modelos de la base de datos y la Application Factory que instancia la aplicación.
- **datasets:** contiene (durante el funcionamiento de la aplicación) todos los conjuntos de datos que los usuarios introducen.
- **anonimos:** conjuntos de datos subidos por los usuarios anónimos.
- **registrados:** conjuntos de datos subidos por los usuarios registrados.
- **seleccionar:** conjuntos de datos para seleccionar de prueba, principalmente durante el desarrollo de la aplicación. También almacena el fichero de prueba que los usuarios pueden descargar.
- **runs:** contiene las ejecuciones de los algoritmos en formato JSON, solo de los usuarios registrados.
- **static:** ficheros estáticos que utiliza la aplicación web: CSS, Javascript, JSON o imágenes.
- **css:** ficheros CSS.
- **js:** ficheros JavaScript.
 - **configuracion:** ficheros JavaScript encargados de la configuración de la ejecución de los algoritmos.
 - **usuarios:** ficheros JavaScript encargados de la generación de tablas y control del espacio personal de los usuarios y el panel de administración.
 - **visualizacion:** ficheros JavaScript encargados de la inicialización y generación de los gráficos en la visualización (gráfico principal y estadísticas).

```
web
├── app
│   ├── static
│   │   ├── json: ficheros JSON.
│   │   ├── pseudocodigos: imágenes de los pseudocódigos.
│   │   │   ├── en: imágenes de los pseudocódigos en inglés.
│   │   │   └── es: imágenes de los pseudocódigos en español.
│   ├── templates: plantillas HTML (Jinja2) que renderiza la
│   │               aplicación web (Flask).
│   ├── configuracion: plantillas concretas de la
│   │                   ventana de configuración
│   │                   de la ejecución de los
│   │                   algoritmos.
│   ├── usuarios: plantillas concretas de las
│   │             distintas ventanas de los
│   │             usuarios: login, registro,
│   │             perfil, espacio personal y
│   │             panel de administración.
│   ├── visualizacion: plantillas concretas de
│   │                   las distintas ventanas
│   │                   de visualización de los
│   │                   algoritmos.
│   ├── translations: traducciones de los textos
│   │                 de la aplicación (por
│   │                 idiomas).
│   ├── tests: pruebas del sistema.
│   │   └── ficheros: ficheros para las pruebas del sistema.
└── instance: contiene la base de datos de la aplicación
              (SQLite).
```

D.3. Manual del programador

El objetivo de este manual es dar el conocimiento necesario al lector/-desarrollador que comience a trabajar con este proyecto para continuarlo. Se ha de tener en cuenta que lo descrito a continuación es lo que se ha utilizado para el desarrollo inicial y será explicado para este (sin limitación, por ejemplo, a usar otras herramientas).

En primer lugar se listan las herramientas consideradas para el desarrollo:

- Python 3.10: Todo el proyecto, desde su inicio, ha sido desarrollado en la versión 3.10.
- Git: Necesario para continuar con el control de versiones del proyecto.
- Pycharm: Editor de código utilizado, podría utilizarse otro si así se considerase.

Python puede descargarse desde su página principal¹. En las herramientas no se ha mencionado «pip» (el administrador de paquetes) pues desde la versión 3.4 de Python este está instalado con él, sin embargo, sería conveniente asegurarse de ello comprobado la versión pues en el futuro será necesario.

Comprobar pip

```
pip --version
python -m ensurepip --upgrade
```

Y de forma general, comprobar que los ficheros binarios pueden ser utilizados por lo menos, en el entorno del programador (solo en la cuenta del usuario o en el equipo completo).

En el caso de Git, desde Linux simplemente se puede instalar con su gestor de paquetes:

Instalar Git en Linux

```
sudo apt install git-all
```

Si el entorno es Windows, existe un instalador directo que puede descargarse desde la página de Git SCM (*Source code management*)².

¹Descargas de Python: <https://www.python.org/downloads/>

²Git para Windows: <https://git-scm.com/download/win>

Pycharm se puede descargar tanto para Windows como para Linux en la página oficial de JetBrains³.

Comprensión de la estructura

Se recomienda leer la sección anterior donde se pueden consultar todos los directorios del proyecto con una breve descripción. La preparación del entorno virtual con el que trabajar se explicará en la próxima sección.

El proyecto se desarrolla en dos ramas comunicadas (de forma unidireccional): los algoritmos implementados y la aplicación web. La aplicación es la que utiliza los algoritmos para obtener la información presentada en la web.

Algoritmos semi-supervisados (contenidos en el directorio `algoritmos`): Los algoritmos desarrollados y nuevos han de situarse en este directorio como raíz. La idea es que cada fichero «.py», homónimo al algoritmo, contenga la definición de un objeto que encapsule el desarrollo del mismo.

Estructura de los objetos (mínima):

Constructor: Donde se configuran los parámetros que necesita el algoritmo. Es recomendable realizar una validación de los mismos por si fueran utilizados de manera individual.

Método de entrenamiento (Fit): Dado que estos algoritmos están pensados no solo para entrenar, sino para almacenar el proceso de entrenamiento y estadísticas, siempre ha de recibir el conjunto de entrenamiento (\mathbf{x} , y), el conjunto de test (\mathbf{x}_{test} , y_{test}) y el nombre de las características de cada instancia (el nombre de las columnas de \mathbf{x}).

En principio, el método de desarrollo seguido es el de primero implementar el algoritmo para después añadir, con la librería Pandas, un registro completo de los momentos de etiquetado y de las estadísticas de cada iteración.

Este método deberá retornar el registro de etiquetado, el registro estadístico y el número de iteraciones realizadas.

Cabecera fit

```
def fit(x, y, x_test, y_test, features)
```

³Pycharm: <https://www.jetbrains.com/pycharm/download/>

Es importante tener en cuenta que esta estructura puede variar en la medida de cómo sea el algoritmo. Por ejemplo, en el caso de *Democratic Co-Learning* se hacía imprescindible añadir estadísticas específicas para cada clasificador que encapsula y por tanto, retornaba más elementos.

Método de predicción: En el caso de algoritmos que trabajan con un único clasificador podría ser opcional, pero en el caso de varios, es necesario considerar cómo se predicen las etiquetas en combinación.

Cabecera predict

```
def predict(self, instances)
```

Métodos estadísticos: Dependiendo de lo que se desee mostrar en la aplicación, se incluirán ciertas estadísticas.

La convención utilizada hasta ahora es crear un método que comience por «get_» seguido del nombre de la estadística, por ejemplo `get_accuracy_score`.

Cabecera ejemplo estadística

```
def get_accuracy_score(self, x_test, y_test):
```

Utilidades En el directorio de las utilidades se han de alojar aquellos métodos que se reutilizan en el proyecto (y que intervenga en algún paso del algoritmo, como por ejemplo, la carga de datos).

Test El desarrollo del software debe ser validado para asegurar su correcto funcionamiento ante las distintas casuísticas. Cuando se desarrolla código en esta sección, sus casos de prueba codificados deben incluirse en el directorio correspondiente. Se utiliza «pytest» como *framework* de pruebas.

Aplicación Web (contenida en el directorio `web`):

La aplicación está desarrollada con el «micro-framework» Flask, que permite la creación de aplicaciones Web en Python. A lo largo del desarrollo la estructura de la aplicación ha variado bastante. Finalmente, se ha modularizado completamente con el uso de **Blueprints** y una **Application Factory** que se encarga de instanciar la aplicación, base de datos y poner en funcionamiento las distintas rutas accesibles.

run.py: Centraliza la ejecución de la aplicación (mediante la **Application Factory**).

instance: Donde se almacena la instancia de la base de datos.

app: Este directorio contiene toda la definición de la aplicación, el resto de los apartados comentados siguientes se encuentran dentro de este.

El fichero `__init__.py` contiene la creación de la aplicación con un método `create_app` (Application Factory). Aquí se especifica toda la configuración, los elementos comunes (como los filtros de Jinja), se instancia la base de datos y se registran las rutas (organizadas con Blueprints como paquetes o extensiones de la aplicación básica).

Blueprints: Las rutas que tiene la aplicación están organizadas en función de su categoría mediante los Blueprints. Y es en los ficheros que terminan por «`_routes.py`» donde se definen. Si se añade una categoría nueva se debe crear un Blueprint que la identifique dentro de su fichero y después debe ser registrado en la aplicación (en `__init__.py`).

Crear Blueprint

```
# El nombre es a elección propia
nuevo_bp = Blueprint('nuevo_bp', __name__)
```

Registrar Blueprint en la aplicación (en `__init__.py`)

```
# Seleccionando el prefijo deseado
app.register_blueprint(nuevo_bp, url_prefix='/')
```

Modelos de la base de datos: En «`models.py`» se tienen definidas todas las entidades que maneja la aplicación: `Users`, `Datasets` y `Runs`. Si se desea añadir alguna, primero se debe crear aquí la definición de la misma. Para que la aplicación sepa las entidades que debe manejar, también es recomendable importarla en la creación de la aplicación.

Importar modelo (en «`create_app()`» de `__init__.py`)

```
from .models import Nueva
```

La **visualización de un algoritmo** se centra en dos pasos: la configuración del mismo (en las rutas `/configuracion/<algoritmo>`) donde se debe renderizar una página con el formulario de configuración (gestionado por `configuration_routes.py`), y la visualización (`/visualizacion/<algoritmo>`) donde se renderiza la página donde se encuentran todos los gráficos de los algoritmos (gestionado por `visualization_routes.py`). Además, existe un paso intermedio en el que se obtienen los datos de la ejecución de los

algoritmos, este paso no es una ruta que pueda visualizarse (gestionado por `data_routes.py`), es un método auxiliar que accede el propio navegador del usuario (petición) para obtener la información. Por lo general, la convención utilizada es que todos los métodos y rutas lleven el nombre del algoritmo.

Las primeras tareas para añadir un algoritmo nuevo serán incorporarlos al flujo de las sesiones y actualizar la plantilla base (por la barra de navegación) y el inicio. La incorporación al flujo de las sesiones se hace incorporándolos a la selección (`/seleccionar/<algoritmo>`), que guarda en la sesión del usuario el nombre del algoritmo.

Formularios: A la hora de crear un nuevo algoritmo, y como se ha comentado, el usuario tiene un paso de configuración del mismo. Esto se realiza mediante un formulario. Para crear uno (WTForm), se debe crear su objeto correspondiente en `forms.py` con los campos necesarios. Hasta el momento, los algoritmos que hay implementados tienen una parte en común que está definida en la clase `FormConfiguracionBase` y en principio, si se añade uno nuevo, deberá extender dicha clase.

Plantillas (templates): Flask utiliza el motor de plantillas Jinja2, que añaden instrucciones en HTML. Obviando que cada algoritmo tiene sus particularidades, están organizadas mediante la extensión de plantillas. De forma general, añadir un algoritmo podría solo conllevar la creación de una plantilla de configuración (incluyendo la creación del formulario) y otra de visualización. En ambos casos se tiene una plantilla base de la que se debe extender. También se deben nombrar como resto de algoritmos (están nombrados de forma intuitiva incluyen el propio nombre del algoritmo).

De forma general, las plantillas se encuentran repartidas en varios subdirectorios, cada uno de ellos engloban una parte de la aplicación.

```
templates: plantillas HTML (Jinja2) que renderiza la
           aplicación Web (Flask).
├── configuracion: plantillas concretas de la ventana de
                   configuración de la ejecución de los
                   algoritmos.
├── usuarios: plantillas concretas de las distintas ventanas
              de los usuarios: login, registro, perfil,
              espacio personal y panel de administración.
└── visualizacion: plantillas concretas de las distintas ventanas
                   de visualización de los algoritmos.
```

Debe evaluarse la adición de nuevas sub-carpetas si así fuera. Además, aquellas plantillas que sean generales y no vinculadas a una parte concreta, se incluirán directamente en `templates`.

Ficheros estáticos (`static`):

El contenido dinámico debe ser creado mediante Javascript, de forma «*vanilla*» (sin utilizar *frameworks*).

Al igual que con las plantillas, es completamente necesario mantener la organización actual de las carpetas, las funciones que se añadan deberán incluirse en la carpeta de su contexto.

```
js: ficheros JavaScript.
├── configuracion: ficheros JavaScript encargados de la
                   configuración de la ejecución de los
                   algoritmos.
├── usuarios: ficheros JavaScript encargados de la
              generación de tablas y control del espacio
              personal de los usuarios y el panel de
              administración.
└── visualizacion: ficheros JavaScript encargados de la
                   inicialización y generación de los gráficos
                   en la visualización (gráfico principal y
                   estadísticas).
```


En menor medida, los estilos deben ser retocados en estos ficheros (`css/style.css`) aunque en principio la aplicación está estilada con Bootstrap 5.

Entre estos ficheros estáticos también se tienen unas imágenes con los pseudocódigos de los algoritmos. Estas imágenes deben verse tanto en la configuración como en la visualización. Se puede consultar alguna de las plantillas existentes de ejemplo.

Otra parte muy importante es si se quieren añadir nuevos clasificadores base con sus parámetros. La especificación de los mismos está almacenada en el fichero JSON «`parametros.json`».

Hasta el momento del desarrollo solo existen dos tipos de entradas, los selectores y los numéricos.

Estructura para añadir clasificadores y sus parámetros

```
{
  "ClasificadorBase": {
    "parametro_numerico": {
      "label": "Nombre a mostrar",
      "type": "number",
      "step": 1,
      "min": 1,
      "max": "Infinity",
      "default": 5
    },
    "parametro_selector": {
      "label": "Nombre a mostrar",
      "type": "select",
      "options": ["lista", "de", "elementos"],
      "default": "elementos"
    }
  }
}
```

Con «*step*» se debe tener en cuenta la posibilidad de permitir números enteros (al introducir 1) o números decimales (al incluir un flotante, por ejemplo, 0.01).

Según se ha codificado, la aparición de este nuevo clasificador será automática, ya que una vez leído este fichero, JavaScript lo recorre creando

tantas opciones en los selectores como clasificadores haya y a su vez, todos los parámetros de estos clasificadores.

Además de esto, hay tres zonas en la lógica de los *endpoints* que deben ser modificadas si se añade un algoritmo. En «`data_routes.py`», «`configuration_routes.py`» y «`visualization_routes.py`» se tiene código específico para cada algoritmo. La «activación» de cada código se realiza mediante un *if* de tal forma que cuando coincide con el nombre del algoritmo, se realizan ciertos pasos concretos para él. Se deberá ajustar la lógica para el nuevo algoritmo.

En «`data_routes.py`» se deberá crear la ruta de datos del algoritmo (que se encarga de instanciar el algoritmo con los parámetros de la configuración, y de ejecutarlo), en «`configuration_routes.py`» es donde se instancian los formularios de los algoritmos y en «`visualization_routes.py`» se obtienen los parámetros introducidos en el formulario mediante una función creada para ello.

Conjunto de datos (*datasets*): Los conjuntos de datos de los usuarios (vinculados a sus sesiones) se almacenan localmente en la aplicación con el «*timestamp*» concatenado al nombre del fichero introducido.

Para agilizar el proceso del mantenimiento (principalmente del almacenamiento del servidor), los ficheros subidos por usuarios anónimos se guardan en una sub-carpeta distinta a la de los usuarios registrados. Con «`current_user`» de Flask-Login se puede conocer si se trata de usuarios anónimos o registrados. En función de esto, la aplicación tiene almacenada una ruta para los ficheros anónimos («`CARPETA_DATASETS_ANONIMOS`») y otra para los registrados («`CARPETA_DATASETS_REGISTRADOS`»).

Internacionalización: La aplicación está pensada para ser internacionalizada en cualquier idioma, aunque solo se ha incluido inglés y español. Una de las herramientas utilizadas es Babel, para incluir nuevo texto en la aplicación se debe utilizar la función `gettext` o `lazy_gettext` (tanto en Jinja2 como Python si fuera necesario). El proceso de extracción y compilación de las traducciones está explicado en la siguiente sección.

Estas funciones envolventes no funcionan en JavaScript. Por si esto ocurre durante el posterior desarrollo, se han considerado dos alternativas.

Para el caso de las visualizaciones, en la plantilla «`base_visualizacion`» se ha incluido una función de traducción embebida directamente en «`<script>`»⁴.

⁴Etiqueta HTML

Al incluirse de esta forma, Babel sí puede capturar esas funciones y al acceder a la función retornar el texto adecuado.

La otra posibilidad es mediante diccionarios y la declaración de una variable que indique el idioma (*locale*) en HTML. Por ejemplo, en el espacio personal (plantilla «miespacio») y en el panel de administración (plantilla «admin»), se ha definido una constante que consulta el idioma actual (es una variable global declarada en Flask con «`@app.context_processor`»). A partir de aquí, cuando existe un texto generado en JavaScript que necesita ser traducido, se define un diccionario que contiene en los valores las palabras deseadas. Este diccionario realmente será un diccionario de diccionarios, el primer nivel tendrá el idioma.

Ejemplo:

```
let diccionario = {  
  "en": {"clave": "palabra en inglés"},  
  "es": {"clave": "palabra en español"}  
};
```

Ahora, cuando un texto ha de ser traducido, primero se accede al primer nivel y luego a la palabra concreta.

```
// locale es la constante declarada en la plantilla  
let palabra_localizada = diccionario[locale]["clave"];
```

D.4. Compilación, instalación y ejecución del proyecto

Para poner en funcionamiento la aplicación primero se debe preparar el entorno de ejecución. Se recuerda la necesidad de tener instalado Python (y el administrador de paquetes «pip»).

Código fuente de la aplicación Para obtener el código fuente de la aplicación, este debe ser descargado del repositorio Github utilizado⁵.

Tanto en Windows como en Linux esto se puede realizar descargando el fichero comprimido de todo el repositorio desde el navegador.

⁵Repositorio: <https://github.com/dma1004/TFG-SemiSupervizado>

Pero si se quiere realizar mediante consola de comandos, y suponiendo que se ha instalado Git:

Clonación de repositorio desde consola

```
$ git clone https://github.com/dma1004/TFG-SemiSupervisado.git
```

La localización del proyecto queda a discreción del programador.

Entorno virtual Python El entorno virtual no es estrictamente necesario aunque es **altamente** recomendable, pues en los próximos pasos se instalarán múltiples librerías que sin entorno virtual quedarían instaladas globalmente. El proceso descrito a continuación puede ser sustituido en caso de que solo se esté trabajando con Pycharm. Este editor permite crear entornos virtuales e incluso realizarlo automáticamente al detectar los distintos requisitos del proyecto. Pero de forma general, suponiendo que se desea preparar un entorno de «pruebas» o «producción» y que la aplicación esté en funcionamiento, se realizan los siguientes pasos:

Creación del entorno virtual (dentro de la carpeta deseado)

```
$ python -m venv ./venv
```

Activación del entorno virtual

```
//Windows  
$ venv\Scripts\activate.bat //CMD  
$ venv\Scripts\Activate.ps1 //PowerShell
```

```
//Linux  
$ source ruta/al/entorno/virtual/bin/activate
```

Para desactivar el entorno (una vez en él)

```
$ deactivate
```

Instalación de paquetes En este paso se van a instalar todas las librerías necesarias, el proyecto trae un fichero «requirements.txt» en el que vienen especificados estos paquetes y sus versiones. Además, los algoritmos implementados también están configurados como un paquete que puede ser instalado.

Instalar librerías externas

```
$ pip install -r requirements.txt
```

Instalar resto de paquetes (definidos en setup.py)

```
$ pip install .
```

Por último, hay tres directorios que deben crearse para almacenar recursos de los usuarios durante la ejecución.

Creación de directorios (desde /web/app)

```
//Windows
$ MD runs
$ MD datasets\anonimos # el directorio datasets ya está creado
$ MD datasets\registrados

-----

//Linux
$ mkdir runs
$ mkdir datasets/anonimos # el directorio datasets ya está creado
$ mkdir datasets/registrados
```

A partir de aquí ya se tiene configurado todo el entorno y los requisitos necesarios para poder ejecutar la aplicación.

Ejecución Referida a la ejecución de la aplicación Web (Flask), se debe estar situado en el directorio **/web** del proyecto.

Ejecución

```
//Windows
$ set FLASK_APP=run.py
$ flask run (--debug)

-----

//Linux
$ flask run (--debug)
```

Existe una última cuestión que no es estrictamente necesaria para funcionar, pero que añade la internacionalización a la aplicación automáticamente.

Internacionalización En el caso de que se haya incluido más texto traducido, este debe ser compilado con Babel para que la aplicación pueda detectarlo.

Proceso de internacionalización (desde /web/app)

```
Extraer los textos encapsulados por gettext
$ pybabel extract -F babel.cfg -o messages.pot .

Extraer los textos encapsulados por lazy_gettext
$ pybabel extract -F babel.cfg -k lazy_gettext -o messages.pot .

Actualizar los textos extraídos en el fichero de compilación
$ pybabel update -i messages.pot -d translations
```

Todos los pasos anteriores son **estrictamente** necesarios, si se realiza una extracción sin incluir «**lazy_gettext**» (por ejemplo) y a continuación se actualizan las traducciones, se perderán aquellas que tenían «**lazy_gettext**».

Una vez que se han extraído los textos, se han de añadir las traducciones (a mano). Todos los textos envueltos por las funciones antes comentadas actúan como identificadores.

En `translations/es/LC_MESSAGES/messages.po` se tienen esos identificadores junto con la traducción (estará vacía si no se ha introducido). En ese fichero se incluirán las traducciones.

El siguiente paso es la compilación de estas traducciones para que Babel pueda sustituir los textos dinámicamente.

Compilación de traducciones (desde /web/app)

```
Compilación de las traducciones
$ pybabel compile -d translations
```

Despliegue en «producción» Los anteriores pasos son el resumen de todo lo que se debe hacer para iniciar la aplicación en un entorno de desarrollo.

Si el programador ha decidido desplegar el proyecto en un servidor y desea ponerlo en producción, es conveniente añadir una capa más a la aplicación. Esta capa consiste en establecer un servidor WSGI (*Web Server Gateway Interface*) que dictará la comunicación entre el servidor web (a discreción del programador) y Flask.

En el proyecto actual este servidor ha sido generado por Gunicorn (*green unicorn*) y su uso es extremadamente simple⁶.

Lanzar servidor WSGI sobre la aplicación

```
//Linux  
$ gunicorn -b :5000 'app:create_app()' (--log-level debug)
```

Esto sería lo mismo que realizar el paso de la ejecución D.4.

Cabe notar que esta es la forma en la que se ha desplegado el proyecto actual (en un entorno Linux) y debe ser tomado como un ejemplo. Gunicorn no puede ser usado en sistemas Windows (el servidor). Una alternativa muy usada para estos sistemas es waitress⁷.

D.5. Pruebas del sistema

En esta sección se van a incluir todas las pruebas realizadas contra la aplicación, con el fin de garantizar que funciona correctamente y hace lo que debería.

Para realizar estas pruebas se ha considerado utilizar Selenium (IDE). Se trata de un entorno de pruebas para aplicaciones Web. Estas pruebas estarán documentadas a continuación y se almacenará su código en el proyecto para poder ejecutarse en cualquier momento (atendiendo a las precondiciones establecidas).

Pruebas con Selenium IDE

Para codificar y ejecutar las pruebas se debe instalar la extensión «Selenium IDE» en el navegador Google Chrome (incorpora funcionalidades que no existen en otros) y activar en la configuración de la extensión de Selenium IDE la opción «Permitir acceso a URL de archivo». Esta opción se utiliza para poder subir archivos. Es más, también se deben configurar las rutas de los ficheros en los pasos en los que estos se cargan (ver descripción de los casos de prueba).

A continuación, se debe abrir el proyecto de pruebas ya creado (incluido en el código fuente). En él aparecerá todo el listado de casos de prueba. Cada uno de ellos puede ser ejecutado por separado o en conjunto (secuencialmente).

⁶Se encuentra entre los requisitos de bibliotecas del proyecto

⁷waitress: <https://pypi.org/project/waitress/>

En principio, se recomienda esta última opción (uno detrás de otro), ya que está preparado para cumplir con las precondiciones.

Como se ha comentado, es **muy importante** configurar correctamente las rutas de los ficheros con respecto al sistema donde se ejecuten.

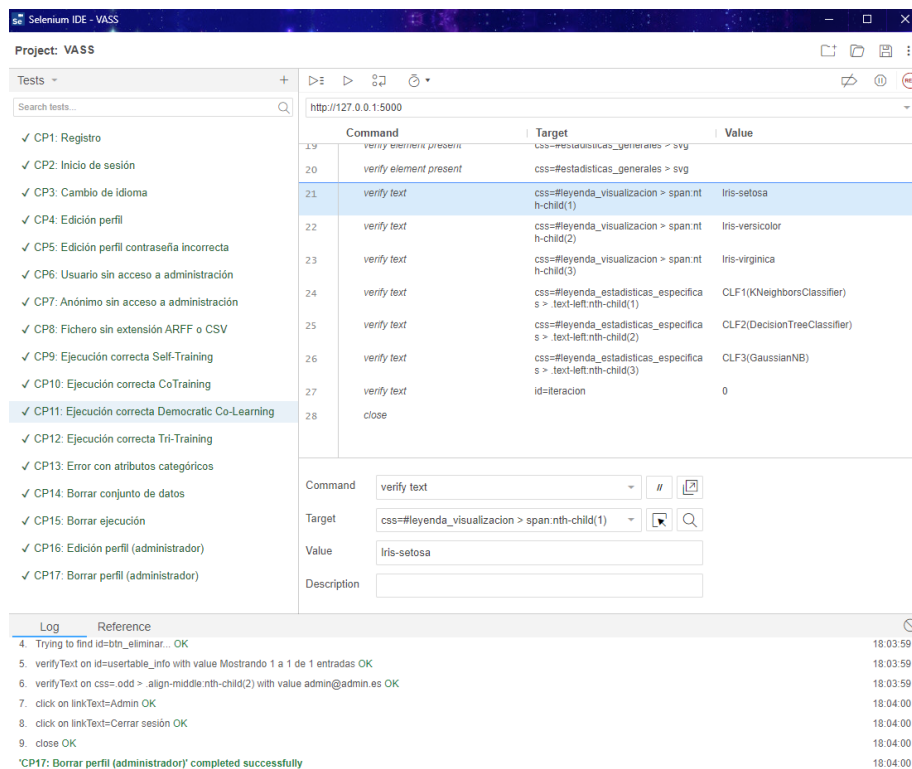


Figura D.1: Entorno Selenium IDE

Casos de prueba

A continuación se presentan todos los casos de prueba que validan el correcto funcionamiento de las partes más importantes de la aplicación.

Algunas de las precondiciones que aparecen a continuación no se han codificado en todas las pruebas de «Selenium», esto es porque los casos de prueba en dicha extensión están pensados para ejecutarse secuencialmente, de tal forma que casos de prueba anteriores pueden dejar la aplicación en el estado correcto para cumplir esas precondiciones.

Durante los siguientes casos de prueba se va a manejar constantemente un usuario de prueba. Este usuario será «Prueba» y tendrá la siguiente información (se hará referencia a él en muchos casos):

- **Nombre:** Prueba
- **Correo electrónico:** prueba@gmail.com
- **Contraseña:** 12345678

Identificador	CP1			
Prioridad	Media			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el registro de usuarios en la aplicación			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	El usuario ha cerrado la sesión			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/?lang=es'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Registrarse»		Encontrarse en la página de registro	Encontrarse en la página de registro	Éxito
Introducir datos de usuario	Nombre: Prueba, Email: prueba@gmail.com, Contraseña: 12345678, Confirmar contraseña: 12345678	Datos introducidos	Datos introducidos	Éxito
Hacer clic en «Enviar»		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Comprobar presencia nombre de usuario		El nombre de usuario «Prueba» se encuentra en la barra de navegación	El nombre de usuario «Prueba» se encuentra en la barra de navegación	Éxito

Tabla D.1: Caso de prueba 1: Registro.

Identificador	CP2			
Prioridad	Media			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el inicio de sesión de usuarios en la aplicación			
Precondiciones	Sin usuario con sesión iniciado y usuario prueba creado			
Postcondiciones	El usuario ha cerrado la sesión			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a ‘/?lang=es’		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Iniciar sesión»		Encontrarse en la página de inicio de sesión	Encontrarse en la página de inicio de sesión	Éxito
Introducir datos de usuario	Nombre: Prueba, Email: prueba@gmail.com, Contraseña: 12345678	Datos introducidos	Datos introducidos	Éxito
Hacer clic en «Enviar»		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Comprobar presencia de nombre de usuario		El nombre de usuario «Prueba» se encuentra en la barra de navegación	El nombre de usuario «Prueba» se encuentra en la barra de navegación	Éxito

Tabla D.2: Caso de prueba 2: Inicio de sesión.

Identificador	CP3			
Prioridad	Baja			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el cambio de idioma en la aplicación			
Precondiciones	Sin precondiciones			
Postcondiciones	Idioma cambiado correctamente			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a ‘/?lang=es’		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Comprobar idioma español		El subtítulo principal es «Trabajo Fin de Grado sobre algoritmos semi-supervisados y su visualización»	El subtítulo principal es «Trabajo Fin de Grado sobre algoritmos semi-supervisados y su visualización»	Éxito
Hacer clic en el icono de idiomas		Menú desplegable de idiomas mostrado	Menú desplegable de idiomas mostrado	Éxito
Hacer clic en el botón de «Inglés»		El subtítulo principal es « <i>Final Degree Project on semi-supervised algorithms and their visualization</i> »	El subtítulo principal es « <i>Final Degree Project on semi-supervised algorithms and their visualization</i> »	Éxito

Tabla D.3: Caso de prueba 3: Cambio de idioma.

Identificador	CP4			
Prioridad	Media			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la edición del perfil de usuario			
Precondiciones	Usuario prueba creado y con sesión iniciada			
Postcondiciones	El usuario prueba tiene los datos originales (para no afectar a casos de prueba futuros)			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/?lang=es'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en el nombre de usuario «Prue- ba» y en el bo- tón de «Perfil»		Encontrarse en la página del perfil de usuario	Encontrarse en la página del perfil de usuario	Éxito
Introducir da- tos de usuario	Nombre: Prueba, Email: prue- bacambia- do@gmail.com, Contrase- ña actual: 12345678	Datos introducidos	Datos introducidos	Éxito
Hacer clic en el botón de «En- viar»		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en el nombre de usuario «Prue- ba» y en el bo- tón de «Perfil»		Encontrarse en la página del perfil de usuario	Encontrarse en la página del perfil de usuario	Éxito
Comprobar email actuali- zado		El email que aparece es «pruebacambia- do@gmail.com»	El email que aparece es «pruebacambia- do@gmail.com»	Éxito

Tabla D.4: Caso de prueba 4: Edición perfil.

Identificador	CP5			
Prioridad	Alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la seguridad de la edición del perfil de usuario			
Precondiciones	Usuario prueba creado y con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a ‘/?lang=es’		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en el nombre de usuario «Prue- ba» y en el bo- tón de «Perfil»		Encontrarse en la página del perfil de usuario	Encontrarse en la página del perfil de usuario	Éxito
Introducir da- tos de usua- rio (contraseña errónea)	Nombre: Prueba, Email: prue- bacambia- do@gmail.com, Contrase- ña actual: 1234567899	Datos introducidos	Datos introducidos	Éxito
Hacer clic en el botón de «En- viar»		El mensaje de error que aparece es «Contraseña actual incorrecta»	El error que apare- ce es «Contraseña actual incorrecta»	Éxito

Tabla D.5: Caso de prueba 5: Edición perfil con contraseña incorrecta.

Identificador	CP6			
Prioridad	Alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la seguridad del acceso a la ruta de administración con usuario registrado			
Precondiciones	Usuario prueba creado y con sesión iniciada			
Postcondiciones	El usuario ha cerrado la sesión			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/admin?lang=es'		Encontrarse en la página de error	Encontrarse en la página de error	Éxito
Comprobar error		El código de error y mensaje son «401» y «No autorizado»	El código de error y mensaje son «401» y «No autorizado»	Éxito

Tabla D.6: Caso de prueba 6: Usuario sin acceso a administración.

Identificador	CP7			
Prioridad	Alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la seguridad del acceso a la ruta de administración con anónimo			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/admin?lang=es'		Encontrarse en la página de inicio de sesión	Encontrarse en la página de inicio de sesión	Éxito
Comprobar advertencia (mensaje flash)		El mensaje es «Por favor, inicia sesión para acceder a esta página.»	El mensaje es «Por favor, inicia sesión para acceder a esta página.»	Éxito

Tabla D.7: Caso de prueba 7: Anónimo sin acceso a administración.

Identificador	CP8			
Prioridad	Alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la denegación de ficheros que no sean ARFF o CSV			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a ‘/?lang=es’		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Self- Training»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir un fichero .txt		Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito
Hacer clic en «Configurar algoritmo»		Redirección a la página de subida	Redirección a la página de subida	Éxito
Comprobar advertencia (mensaje flash)		El mensaje es «El fichero debe ser ARFF o CSV»	El mensaje es «El fichero debe ser ARFF o CSV»	Éxito

Tabla D.8: Caso de prueba 8: Fichero sin extensión ARFF o CSV.

Identificador	CP9			
Prioridad	Muy alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la ejecución del algoritmo Self-Training			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Self-Training»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir el conjunto de datos iris	Fichero «iris.arff»	Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito
Hacer clic en «Configurar algoritmo»		Redirección a la página de configuración	Redirección a la página de configuración	Éxito
Seleccionar «class» como atributo de la clase				Éxito
Hacer clic en «Ejecutar»		Encontrarse en la página de visualización	Encontrarse en la página de visualización	Éxito
Comprobar la existencia de los «SVGs»		Se han generado los «SVGs» (visualización y estadísticas generales)	Se han generado los «SVGs» (visualización y estadísticas generales)	Éxito
Comprobar leyenda principal		La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	Éxito
Comprobar iteración 0		El valor de la iteración es 0	El valor de la iteración es 0	Éxito

Tabla D.9: Caso de prueba 9: Ejecución correcta Self-Training.

Identificador	CP10			
Prioridad	Muy alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la ejecución del algoritmo Co-Training			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Co-Training»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir el conjunto de datos iris	Fichero «iris.arff»	Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito
Hacer clic en «Configurar algoritmo»		Redirección a la página de configuración	Redirección a la página de configuración	Éxito
Seleccionar «KNeighborsClassifier» y «DecisionTreeClassifier» clasificadores (en ese orden)				Éxito
Seleccionar «class» como atributo de la clase				Éxito
Hacer clic en «Ejecutar»		Encontrarse en la página de visualización	Encontrarse en la página de visualización	Éxito
Comprobar la existencia de los «SVGs»		Se han generado los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Se han generados los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Éxito

Comprobar leyenda principal	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	Éxito
Comprobar leyenda estadísticas específicas	La leyenda incluye: «CLF1 (KNeighborsClassifier)» y «CLF2 (DecisionTreeClassifier)»	La leyenda incluye: «CLF1 (KNeighborsClassifier)» y «CLF2 (DecisionTreeClassifier)»	Éxito
Comprobar iteración 0	El valor de la iteración es 0	El valor de la iteración es 0	Éxito

Tabla D.10: Caso de prueba 10: Ejecución correcta Co-Training.

Identificador	CP11			
Prioridad	Muy alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la ejecución del algoritmo Democratic Co-Learning			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Democratic Co-Learning»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir el conjunto de datos iris	Fichero «iris.arff»	Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito
Hacer clic en «Configurar algoritmo»		Redirección a la página de configuración	Redirección a la página de configuración	Éxito
Seleccionar «KNeighborsClassifier», «DecisionTreeClassifier» y «GaussianNB» clasificadores (en ese orden)				Éxito
Seleccionar «class» como atributo de la clase				Éxito
Hacer clic en «Ejecutar»		Encontrarse en la página de visualización	Encontrarse en la página de visualización	Éxito

Comprobar la existencia de los «SVGs»	Se han generado los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Se han generado los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Éxito
Comprobar leyenda principal	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	Éxito
Comprobar leyenda estadísticas específicas	La leyenda incluye: «CLF1 (KNeighborsClassifier)», «CLF2 (DecisionTreeClassifier)», «CLF3 (GaussianNB)»	La leyenda incluye: «CLF1 (KNeighborsClassifier)», «CLF2 (DecisionTreeClassifier)», «CLF3 (GaussianNB)»	Éxito
Comprobar iteración 0	El valor de la iteración es 0	El valor de la iteración es 0	Éxito

Tabla D.11: Caso de prueba 11: Ejecución correcta Democratic Co-Learning.

Identificador	CP12			
Prioridad	Muy alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar la ejecución del algoritmo Tri-Training			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Tri-Training»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir el conjunto de datos iris	Fichero «iris.arff»	Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito

Hacer clic en «Configurar algoritmo»	Redirección a la página de configuración	Redirección a la página de configuración	Éxito
Seleccionar «KNeighborsClassifier», «DecisionTreeClassifier» y «GaussianNB» clasificadores (en ese orden)			Éxito
Seleccionar «class» como atributo de la clase			Éxito
Hacer clic en «Ejecutar»	Encontrarse en la página de visualización	Encontrarse en la página de visualización	Éxito
Comprobar la existencia de los «SVGs»	Se han generado los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Se han generado los «SVGs» (visualización, estadísticas generales y estadísticas específicas)	Éxito
Comprobar leyenda principal	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	La leyenda incluye: «Iris-setosa», «Iris-versicolor», «Iris-virginica»	Éxito
Comprobar leyenda estadísticas específicas	La leyenda incluye: «CLF1 (KNeighborsClassifier)», «CLF2 (DecisionTreeClassifier)», «CLF3 (GaussianNB)»	La leyenda incluye: «CLF1 (KNeighborsClassifier)», «CLF2 (DecisionTreeClassifier)», «CLF3 (GaussianNB)»	Éxito
Comprobar iteración 0	El valor de la iteración es 0	El valor de la iteración es 0	Éxito

Tabla D.12: Caso de prueba 12: Ejecución correcta Tri-Training.

Identificador	CP13			
Prioridad	Alta			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el control de errores de la condición de entrada sobre los atributos categóricos			
Precondiciones	Sin usuario con sesión iniciada			
Postcondiciones	Sin postcondiciones			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a ‘/?lang=es’		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en «Self-Training»		Encontrarse en la página de subida	Encontrarse en la página de subida	Éxito
Subir el conjunto de datos iris	Fichero «iris.arff»	Porcentaje de subida al 100 %	Porcentaje de subida al 100 %	Éxito
Hacer clic en «Configurar algoritmo»		Redirección a la página de configuración	Redirección a la página de configuración	Éxito
Seleccionar «petalwidth» como atributo de la clase				Éxito
Hacer clic en «Ejecutar»		Encontrarse en la página de visualización	Encontrarse en la página de visualización	Éxito
Esperar a la aparición de una ventana emergente				Éxito
Comprobar mensaje de aviso		Mensaje: «Se han detectado características categóricas o indefinidas, recuerde que los algoritmos solo soportan características numéricas»	Mensaje: «Se han detectado características categóricas o indefinidas, recuerde que los algoritmos solo soportan características numéricas»	Éxito

Tabla D.13: Caso de prueba 13: Error con atributos categóricos.

Identificador	CP14			
Prioridad	Media			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el borrado de un conjunto de datos (fichero)			
Precondiciones	Usuario prueba creado, con sesión iniciada y con un único fichero subido			
Postcondiciones	Conjunto de datos borrado			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/?lang=es'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en el nombre de usuario «Prueba» y en el botón de «Mi Espacio»		Encontrarse en la página del espacio personal	Encontrarse en la página del espacio personal	Éxito
Hacer clic en el botón de eliminar (símbolo de papelera) en la tabla de los conjuntos de datos		Modal de confirmación visible	Modal de confirmación visible	Éxito
Hacer clic en «Eliminar» (en el modal)		Modal ocultado y tabla de conjunto de datos vacía	Modal ocultado y tabla de conjunto de datos vacía	Éxito

Tabla D.14: Caso de prueba 14: Borrar conjunto de datos.

Identificador	CP15			
Prioridad	Media			
Fecha de ejecución	29/05/2023			
Objetivo	Probar el borrado de una ejecución del historial			
Precondiciones	Usuario prueba creado, con sesión iniciada y con una única ejecución anterior			
Postcondiciones	Ejecución borrada y sin sesión iniciada			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/?lang=es'		Encontrarse en la página principal	Encontrarse en la página principal	Éxito
Hacer clic en el nombre de usuario «Prue- ba» y en el bo- tón de «Mi Es- pacio»		Encontrarse en la página del espacio personal	Encontrarse en la página del espacio personal	Éxito
Hacer clic en el botón de elimi- nar (símbolo de papelera) en la tabla del histo- rial de ejecucio- nes		Modal de confirma- ción visible	Modal de confirma- ción visible	Éxito
Hacer clic en «Eliminar» (en el modal)		Modal ocultado y tabla del historial de ejecuciones	Modal ocultado y tabla del historial de ejecuciones	Éxito

Tabla D.15: Caso de prueba 15: Borrar ejecución.

Identificador	CP16			
Prioridad	Alta			
Fecha de ejecución	30/05/2023			
Objetivo	Probar la edición del perfil de un usuario por parte de un administrador			
Precondiciones	Usuario prueba como único usuario creado (aparte del administrador) y administrador con sesión iniciada			
Postcondiciones	El usuario prueba tiene el email modificado			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/admin?lang=es'		Encontrarse en la página de administración	Encontrarse en la página de administración	Éxito
Hacer clic en el botón de edición (símbolo de lápiz) del usuario «Prueba» en la tabla de usuarios		Encontrarse en la página de edición	Encontrarse en la página de edición	Éxito
Introducir datos de usuario	Nombre: Prueba, Email: pruebaadministrador@gmail.com, Contraseña actual: 12345678	Datos introducidos	Datos introducidos	Éxito
Hacer clic en el botón de «Enviar»		Encontrarse en la página de administración	Encontrarse en la página de administración	Éxito
Comprobar email actualizado del usuario «Prueba» en la tabla de usuarios		El email que aparece es «pruebaadministrador@gmail.com»	El email que aparece es «pruebaadministrador@gmail.com»	Éxito

Tabla D.16: Caso de prueba 16: Edición perfil (administrador).

Identificador	CP17			
Prioridad	Alta			
Fecha de ejecución	30/05/2023			
Objetivo	Probar el borrado de un usuario por parte de un administrador			
Precondiciones	Usuario prueba como único usuario creado (aparte del administrador) y administrador con sesión iniciada			
Postcondiciones	El usuario ha sido borrado			
Paso	Entrada	Resultado esperado	Resultado obtenido	Estado
Acceder a '/admin?lang=es'		Encontrarse en la página de administración	Encontrarse en la página de administración	Éxito
Hacer clic en el botón de eliminar (símbolo de papelera) del usuario «Prueba» en la tabla de usuarios		Modal de confirmación visible	Modal de confirmación visible	Éxito
Hacer clic en «Eliminar» (en el modal)		Modal ocultado y tabla de usuario con una única entrada	Modal ocultado y tabla de usuario con una única entrada	Éxito
Comprobar que la única entrada es la del administrador		La entrada corresponde con el email «admin@admin.es»	La entrada corresponde con el email «admin@admin.es»	Éxito

Tabla D.17: Caso de prueba 17: Borrar perfil (administrador).

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección se presenta los requisitos que el usuario debe satisfacer para utilizar la aplicación junto con la instalación y manual de la misma.

E.2. Requisitos de usuarios

Los requisitos que debe cumplir el usuario son:

- Periféricos básicos: pantalla, teclado y ratón.
- Navegador Web (Firefox, Chrome, Edge...).
- Conexión a internet.
- JavaScript habilitado en el navegador.

E.3. Instalación

El usuario no necesita instalar el software para utilizarlo. Puede acceder a él directamente desde su navegador en <https://vass.dmachacha.dev>.

E.4. Manual del usuario

Antes de iniciar el manual, se ha creado un usuario administrador (igual que un usuario registrado pero con más privilegios) para probar la aplicación.

Credenciales administrador

- Email: admin@admin.es
- Contraseña: 12345678

Se ha creado una biblioteca de videotutoriales con los pasos más importantes de la aplicación, tanto para demostración como aprendizaje. El tiempo de visualización (en conjunto) es de unos 8 minutos.

Acceso a los videotutoriales

Puede accederse directamente desde el [repositorio](#).
Puede accederse mediante [OneDrive](#).

Con la ayuda de imágenes capturadas directamente de la web, esta sección describe cómo se realizan todas las acciones de la aplicación.

Dado que la documentación presente se encuentra en español, todas las interfaces se mostrarán en español. Aun así, la aplicación está preparada para el idioma inglés de igual forma.

Este manual está pensado para los usuarios anónimos y los usuarios con cuenta registrada.

Visualizar un algoritmo

El flujo para visualizar un algoritmo es el siguiente:



Figura E.1: Flujo de la visualización de un algoritmo

Seleccionar algoritmo

Para seleccionar un algoritmo, se puede hacer de dos formas. La primera es haciendo clic a los enlaces que aparecen en la barra de navegación (que además está siempre presente en todas las pestañas) y también haciendo clic en las tarjetas de presentación de cada algoritmo en la página principal (ver figura [E.2](#)).

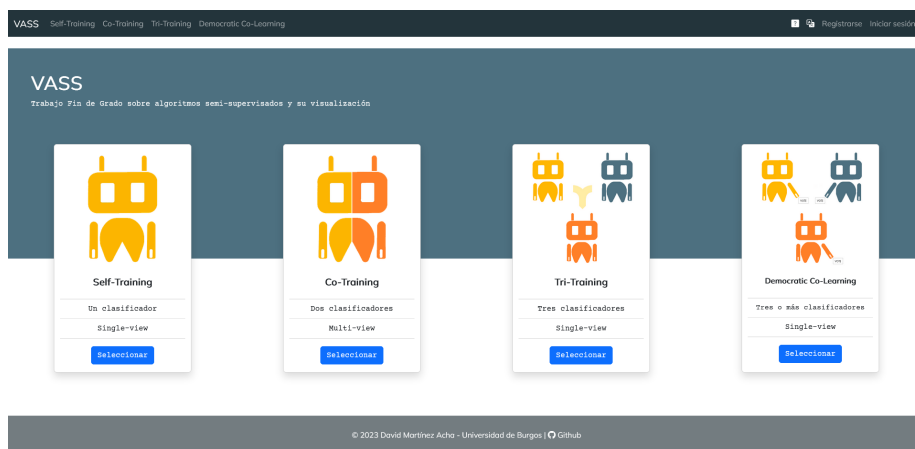


Figura E.2: Página principal

Una vez que se haya seleccionado, será redirigido a la página de subida del fichero.

Carga del conjunto de datos

El fichero contendrá el conjunto de datos. Para subir un fichero, simplemente se puede arrastrar desde el propio sistema hasta la zona marcada con rayas o abriendo el explorador de archivos con el botón de «Selecciona fichero» (ver figura E.3). Podrá ver durante la carga el progreso de la misma.

Si el usuario no dispone de un fichero, la aplicación incluye un enlace para descargar uno de prueba, pulsando en el botón «Descargar fichero de prueba». A partir de aquí, es el mismo procedimiento comentado anteriormente.

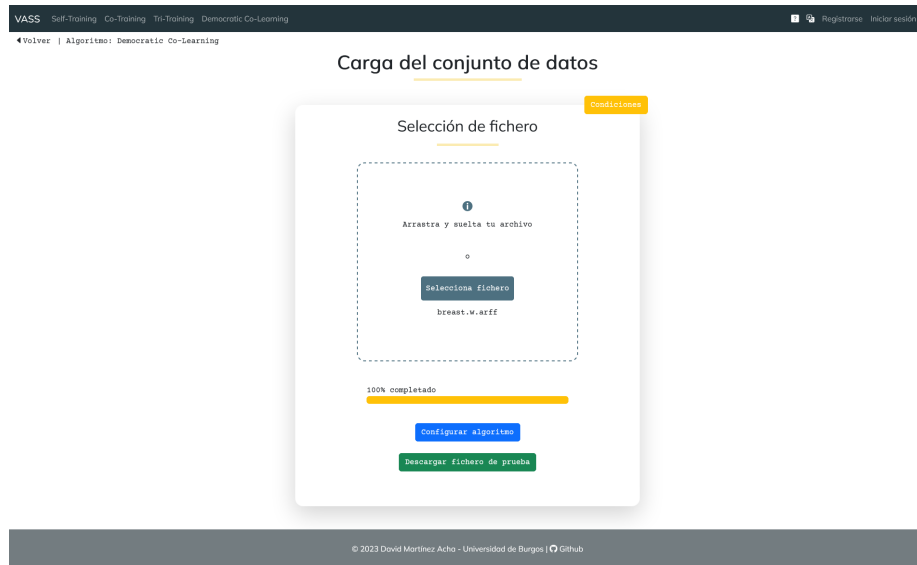


Figura E.3: Carga del conjunto de datos

Consideraciones del conjunto de datos En primer lugar, los ficheros subidos solo podrán tener extensiones ARFF o CSV, en caso contrario, al intentar pasar al siguiente paso, el usuario será devuelto a esta misma página con un mensaje de error.

El contenido del fichero de datos tiene que cumplir un requisito fundamental derivado de la ausencia de un pre-procesado completo:

Requisito fundamental

Todos los atributos del conjunto de datos deben ser numéricos (internamente los algoritmos requieren de este tipo de datos), esto **no** incluye al atributo de la clase, que sí puede ser categórico/nominal (esa parte del pre-procesado sí que es realizada).

Además, si el conjunto de datos es semi-supervisado, este debe tener -1, -1.0 o «?» en los datos no etiquetados. Si en un dato aparece un -1 el resto de no etiquetados deben ser también -1.

Para el caso de ARFF se debe tener en cuenta su propio formato. Por ejemplo, si la clase está declarada con varios valores, las etiquetas tienen que ser uno de esos valores. En este sentido, si se quiere indicar un no etiquetado (o desconocido) es donde se incorpora «?» (ARFF permite este símbolo incluso cuando no es un valor declarado en la lista de posibles valores).

Las condiciones de entrada vienen especificadas al pulsar en el botón de la esquina superior derecha del recuadro de selección de fichero «Condiciones». Al pulsar en él aparecerá una ventana emergente con dichas condiciones (ver figura E.4).

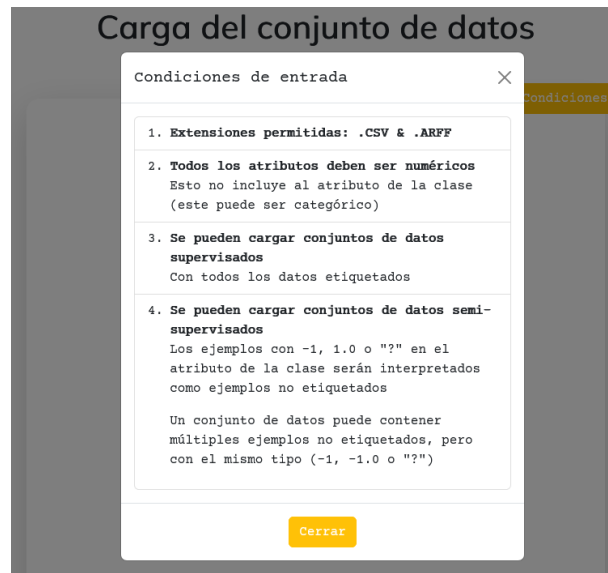


Figura E.4: Condiciones de entrada

Es importante remarcar que salvo la condición de la extensión, la aplicación no puede controlar en este punto el resto de condiciones (ocurrirán mensajes de error en durante el manejo posterior).

Una vez que el fichero ha sido cargado (porcentaje completado), se habrá habilitado el botón de configuración. Pulsando en él, se redirigirá a la siguiente página del flujo (configuración).

Configuración del algoritmo

Se encontrará en la página de configuración del algoritmo, donde podrá observar un apartado teórico con sus conceptos generales y su pseudocódigo. Por otro lado, tendrá un formulario con todos los parámetros que se pueden configurar para ese algoritmo (ver figura E.5).

Configuración del algoritmo: Self-Training

Teoría

Se trata del método de aprendizaje semi-supervisado más sencillo y "directo". Este método "envuelve" un único clasificador base, que entrena con los datos etiquetados iniciales y aprovecha el proceso de pseudo-etiquetado para continuar su entrenamiento. El método comienza por entrenar ese clasificador con los datos etiquetados que se tienen. A partir de este aprendizaje inicial, se etiqueta el resto de datos. De todas las nuevas predicciones se seleccionan aquellas en las que el clasificador más confianza tiene (de haber acertado). Una vez seleccionados, el clasificador es reentrenado con la unión de los ya etiquetados y estos nuevos. El proceso continúa hasta un criterio de parada (generalmente hasta etiquetar todos los datos o un número máximo de iteraciones).

Algoritmo Self-Training

Input: Conjunto de datos etiquetados L , no etiquetados U y clasificador H

Output: Clasificador entrenado

```

1 while  $|U| \neq 0$ 
2   Entrenar  $H$  con  $L$ 
3   Predecir etiquetas de  $U$ 
4   Seleccionar un conjunto  $T$  con aquellos datos que tenga la mayor probabilidad
5    $L = L \cup T$ 
6    $U = U - T$ 
7 endwhile
8 Entrenar  $H$  con  $L$ 
9 return  $H$ 

```

Parámetros

Selecciona clasificador: SVC

Selecciona el atributo de la clase/etiqueta del conjunto de datos:

Selecciona la componente X: Clump_Thickness

Selecciona la componente Y: Clump_Thickness

Utilizar PCA para reducir a 2D: ☒

Estandarizar: ☒

Porcentaje de no etiquetados: %

Porcentaje de test (después de aplicar el no etiquetados): %

Limita: %

Número de iteraciones:

Ejecutar

© 2023 David Martínez Acha - Universidad de Burgos

Figura E.5: Configuración del algoritmo

Todos los parámetros tienen establecido un valor por defecto (con la configuración «estable»), pero se tiene libertad completa para modificar cada uno de ellos. En principio, si solamente se quiere ejecutar sin modificar ningún parámetro, es necesario seleccionar correctamente el atributo de la clase. Este atributo no puede ser establecido automáticamente por la aplicación, ya que depende del conjunto de datos subido.

Una vez configurado, se puede visualizar el algoritmo pulsando en el botón de «Ejecutar».

Visualización

Ya en la página de visualización, se mostrará durante unos momentos una animación de carga. Cuando el sistema haya finalizado la ejecución, se podrán ver los distintos gráficos (ver figura E.6).

En principio, los errores que ocurran en el sistema serán mostrados. Sin embargo, en el caso de que la animación dure un periodo de tiempo excesivo, se recomienda reintentar la configuración (podrían ser problema de red simplemente).

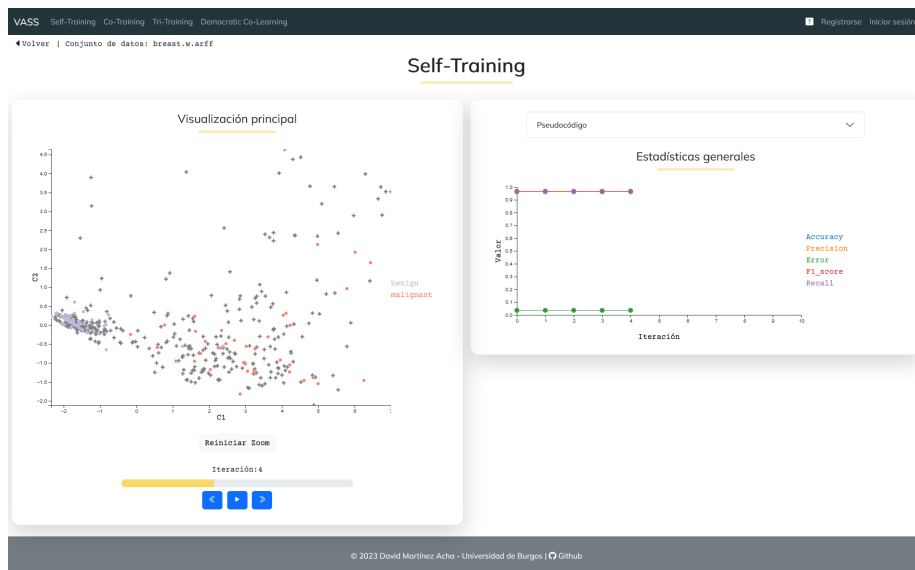


Figura E.6: Vista general de una visualización

Visualización principal En el gráfico principal se mostrará el conjunto de datos en dos dimensiones. Aquí podrá verse qué es lo que ocurre durante el proceso de entrenamiento del algoritmo (ver figura E.7).

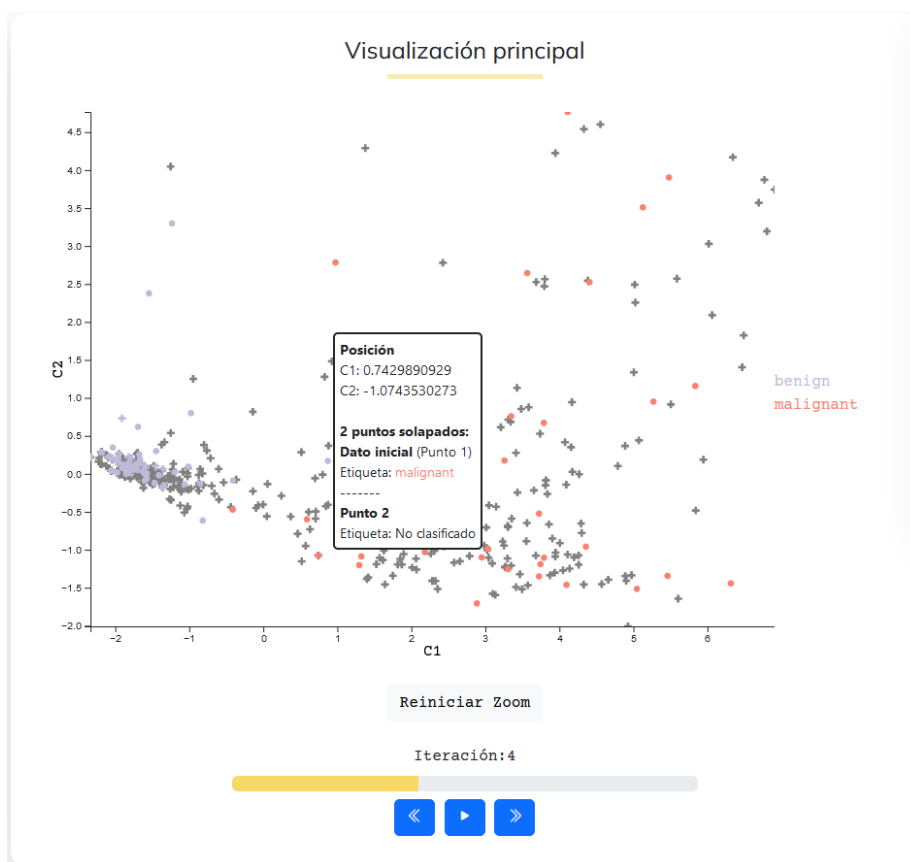


Figura E.7: Visualización principal

Este gráfico es interactivo:

- Permite realizar **zoom** sobre una zona deseada mediante el doble clic o moviendo la ruleta del ratón.
- Al pasar el ratón por uno de los puntos, se mostrará toda la información relativa a esa posición en un recuadro informativo (aparecerá en las proximidades del ratón). Esto se está ejemplificando en la imagen anterior.

Para controlar la evolución, en la parte inferior se encuentra un panel de control que permite:

- Reiniciar **zoom**. Pese a que es posible reducir/aumentar el **zoom** manualmente, sirve para volver a la posición original.

- Visualizar la iteración actual: mediante el número y una barra de progreso.
- Reproducir automáticamente pulsando en el botón con el símbolo *play*.
- Avanzar iteración manualmente.
- Reducir iteración.

Todas estas acciones modificarán el estado de los gráficos.

Tooltip El *tooltip* que se muestra al pasar el ratón por encima de un punto tiene varios formatos dependiendo del algoritmo mostrado y de los datos introducidos.

Tooltip: Casos comunes Existe un formato común a todos los algoritmos para los casos de los datos iniciales. Este tipo de puntos se representa mediante un círculo. Como dato inicial, tendrá la etiqueta correspondiente. Además, cada punto tiene en la parte superior la posición que ocupa en el gráfico. Esta posición coincide con los atributos seleccionados para representar los datos. En la figura E.8 se presenta un ejemplo de todo lo anterior donde se seleccionó PCA y por eso aparece «C1» y «C2».

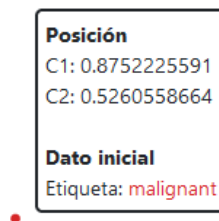


Figura E.8: Tooltip con dato inicial

Otro formato común es en el caso de puntos solapados (derivados de ejemplos duplicados en el conjunto de datos o por PCA). En este caso, donde en el anterior aparecía la información del punto, ahora aparecerá un listado con todos los puntos solapados (ver figura E.9).

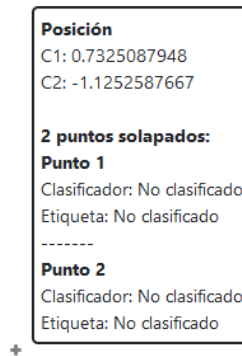


Figura E.9: Tooltip con datos solapados

Como se puede ver en este ejemplo de la figura E.9, los puntos no han sido clasificados (captura tomada en iteración cero), esto no es importante, la diferencia con una iteración posterior es que aparecerá la etiqueta asignada (se verá en cada algoritmo). Además, aparece un indicativo de «Clasificador», se ha querido incluir en este ejemplo porque en todos los algoritmos excepto Self-Training se indica el clasificador que se ha encargado de etiquetar cada punto.

Es interesante comentar que puede darse el caso de puntos solapados en el que alguno o todos sean datos iniciales (ver figura E.10).

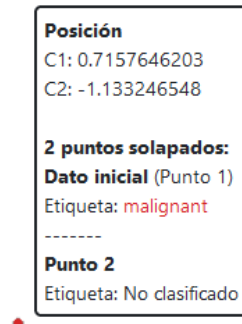


Figura E.10: Tooltip con datos solapados con uno inicial

Tooltip: Self-Training El formato de tooltip para Self-Training no tiene grandes complicaciones y comprendiendo los anteriores ejemplos resulta sencillo de interpretar.

Existen dos formatos, el primero es en el que el dato no ha sido etiquetado (porque se etiqueta en una iteración posterior o simplemente porque nunca es etiquetado). Por ejemplo, una captura tomada en la iteración dos de

una dato no etiquetado es la de la figura E.11. En dicho ejemplo puede ser extraño que no aparezca un título indicando el punto (algo como «Punto 1»), ese formato solo se «activa» cuando existen datos solapados.

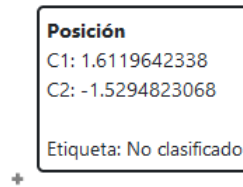


Figura E.11: Tooltip con un dato no clasificado

El otro formato es en el que el dato ya ha sido clasificado (en la iteración actual o en una previa). Por ejemplo, una captura tomada en la iteración ocho de una dato etiquetado en la iteración seis es la de la figura E.12.

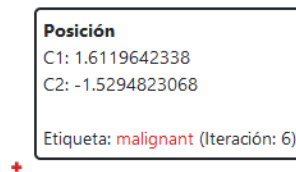


Figura E.12: Tooltip con un dato clasificado

En este formato anterior se muestra la etiqueta asignada así como la iteración en la que se clasificó entre paréntesis.

Tooltip: Co-Training Realmente, el tooltip para Co-Training es muy similar a Self-Training. No existe ningún caso extraño o adicional.

Vuelven a existir dos formatos, cuando el dato no ha sido etiquetado todavía (o nunca, ver figura E.13) y cuando sí está etiquetado.

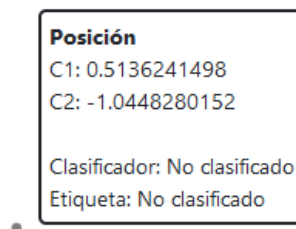


Figura E.13: Tooltip con un dato no clasificado

Para el caso del dato clasificado sí que es necesario puntualizar alguna cuestión (se verá en el ejemplo siguiente). Co-Training considera dos clasificadores base y cada uno de ellos puede clasificar a un punto. Esto se ha representado de dos maneras. La primera señal es que el símbolo del punto será uno concreto para cada clasificador. La segunda señal es que en el *tooltip* aparecerá una línea de «Clasificador:» que estará seguida de dicho símbolo y el nombre del clasificador (ver figura E.14). Se mantiene también el número de la iteración en la que se ha clasificado.



Figura E.14: Tooltip con un dato clasificado

Esta idea también es la misma si existen datos solapados. En ese caso, en cada uno de los puntos que se indiquen en el listado de solapados aparecerá el clasificador (el símbolo y el nombre) junto con la etiqueta (ver figura E.15).

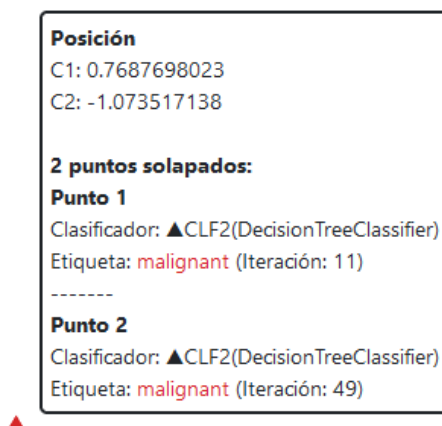


Figura E.15: Tooltip con datos solapados y clasificados

Tooltip: Democratic Co-Learning y Tri-Training En el caso de que un dato no haya sido clasificado es exactamente lo mismo que para Co-Training, simplemente mostrará la posición y que no ha sido clasificado («No clasificado»).

La particularidad que tiene Democratic Co-Learning y Tri-Training es que cada dato puede ser etiquetado por varios clasificadores. Para ejemplificar esto simplemente se realiza un listado de los clasificadores que lo han etiquetado (ver figura E.16).

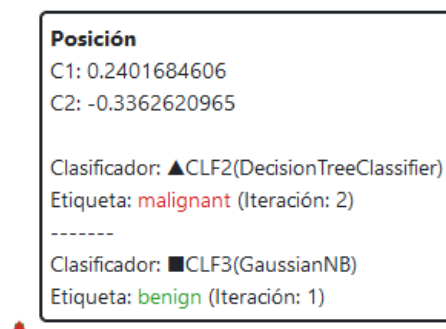


Figura E.16: Tooltip con un dato clasificado por dos clasificadores

Es de destacar que solo aparecen aquellos que en la iteración actual o una previa han etiquetado el dato.

El formato de ambos algoritmos es el mismo. Sin embargo, se cree conveniente explicar el funcionamiento básico.

En **Democratic Co-Learning** cada punto solo puede ser clasificado **una vez** por cada clasificador durante toda la ejecución. Esto, a efectos de visualización, significa que el listado de clasificadores del tooltip solo puede aumentar (si es que el punto es clasificado alguna vez).

En **Democratic Co-Learning** cada punto puede ser clasificado **varias veces** por cada clasificador. Esto es así porque cada clasificador mantiene su propio conjunto de entrenamiento y este es vaciado al comienzo de cada iteración (y rellenado durante la misma). Todo esto significa que durante una visualización, un ejemplo puede ser clasificado por una lista de clasificadores y en la siguiente iteración por otra (igual o distinta).

Esto no afecta al formato comentado, pero puede llegar a ser extraño sin especificarlo.

Por último, como en todos los anteriores formatos, pueden existir datos solapados. El ejemplo mostrado en la figura E.17 es de Democratic Co-Learning, pero como se ha comentado, es el mismo formato que el de Tri-Training.

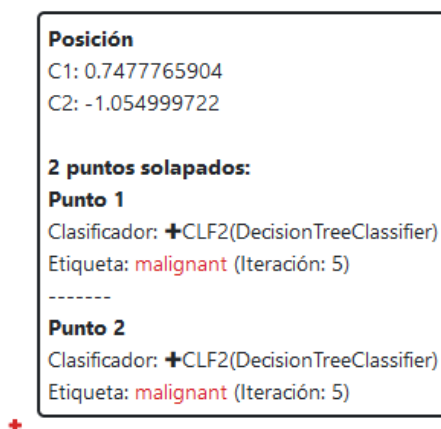


Figura E.17: Tooltip con datos solapados y clasificados

En este ejemplo solo un clasificador ha clasificado cada dato solapado. Se ha elegido este ejemplo porque, como es de esperar, si los tres clasificadores etiquetan, el *tooltip* crecerá de igual manera.

Gráficos estadísticos Pasando a otra parte de la ventana completa de visualización, en la zona de la derecha se incluirá el resto de gráficos adicionales, que serán principalmente estadísticas.

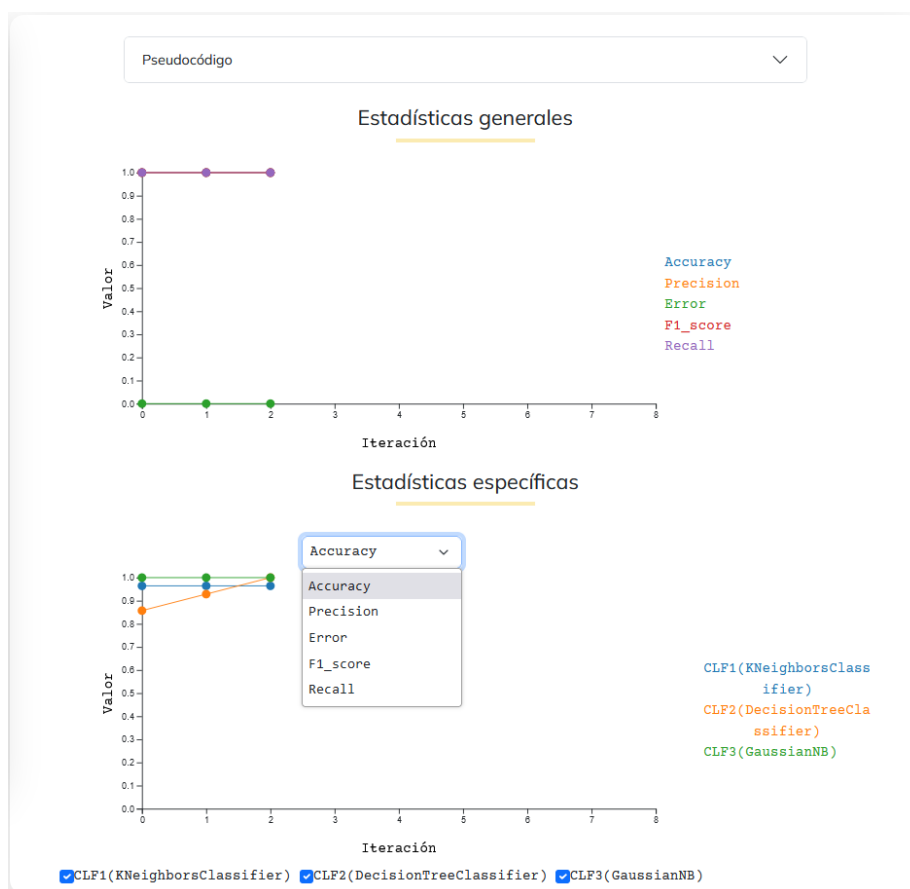


Figura E.18: Gráficos estadísticos

En la parte superior de esta zona se tiene un desplegable (contraído por defecto) que contiene el pseudocódigo (el mismo que en la fase de configuración). Si se desea consultar, simplemente se pulsa en cualquier parte del desplegable (ver figura E.19).

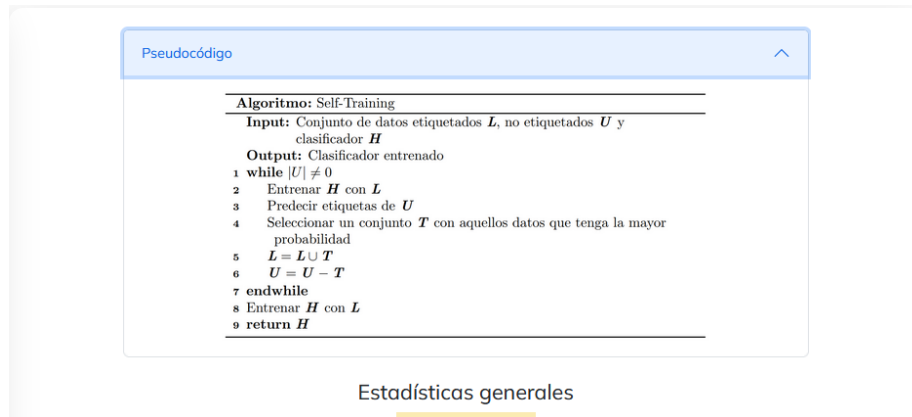


Figura E.19: Desplegable pseudocódigo

El gráfico de estadísticas generales simplemente será a interpretación del usuario (no puede realizar ninguna opción).

En el caso de las estadísticas específicas puede seleccionar qué estadística mostrar mediante el selector superior, y de qué clasificadores mostrarla mediante las casillas en la parte inferior (ver figura E.18).

Ambos gráficos anteriores contienen información similar, en el eje X se indican las iteraciones y en el eje Y el valor de la estadística(s).

Ayuda y cambio de idioma

Aunque la propia aplicación detecta el idioma más adecuado (entre español e inglés) que mostrar, se puede seleccionar el idioma de forma manual.

En la barra de navegación hay un símbolo de traducción característico que al pulsar aparece un desplegable.

También hay un icono con una interrogación, este es un enlace directo al manual de usuario de la aplicación (esta documentación). Al pulsar en él se abre una nueva pestaña con dicho manual (sin modificar la actual y sin descargar).

Ambas funciones pueden verse en la figura E.20.

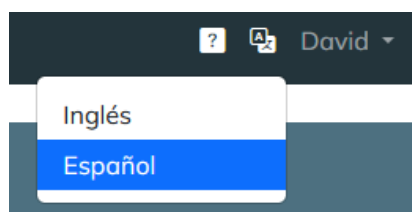


Figura E.20: Cambiar de idioma

Pulsando en el idioma se refrescará la página acorde al idioma seleccionado. Esta característica **no** se incluye en la página de visualización.

Registrarse

Para aquellos usuarios anónimos que deseen crear una cuenta en la aplicación, será necesario realizar el proceso de registro.

Para acceder a él, en la barra de navegación se dispone de un enlace (en la parte derecha) que redirecciona al formulario de creación (ver figura E.21). Una vez accedido, se deben rellenar los siguientes campos (todos obligatorios):

- Nombre: entre 2 y 10 caracteres.
- Correo electrónico: será el identificador del usuario en el sistema y por lo tanto solo podrá haber uno.
- Contraseña: con al menos ocho caracteres.
- Confirmar contraseña: misma contraseña que el campo anterior.

The screenshot shows the 'Registrarse' form within the VASS application. The top navigation bar includes links for 'VASS', 'Self-Training', 'Co-Training', 'Tri-Training', and 'Democratic Co-Learning', along with icons for user actions and links to 'Registrarse' and 'Iniciar sesión'. The form itself is titled 'Registrarse' and contains four input fields: 'Nombre', 'Correo electrónico', 'Contraseña', and 'Confirmar contraseña'. A blue 'Enviar' button is positioned at the bottom of the form. The footer of the page displays the copyright notice '© 2023 David Martínez Acha - Universidad de Burgos' and a GitHub logo.

Figura E.21: Formulario de registro

Una vez enviado el formulario (y después de la comprobación de todos los campos), la cuenta quedará registrada y se habrá iniciado sesión automáticamente.

Iniciar sesión

Al igual que para el registro, para iniciar sesión existe un enlace en la barra de navegación (en la parte derecha) que redirecciona al formulario de inicio de sesión (ver figura E.22).

The screenshot shows the 'Iniciar sesión' form within the VASS application. The top navigation bar is identical to the previous figure, with links for 'VASS', 'Self-Training', 'Co-Training', 'Tri-Training', and 'Democratic Co-Learning', along with icons for user actions and links to 'Registrarse' and 'Iniciar sesión'. The form is titled 'Iniciar sesión' and contains two input fields: 'Correo electrónico' and 'Contraseña'. A blue 'Enviar' button is at the bottom. A link labeled 'Crear cuenta' is located above the 'Enviar' button. The footer of the page displays the copyright notice '© 2023 David Martínez Acha - Universidad de Burgos' and a GitHub logo.

Figura E.22: Formulario de inicio de sesión

Este formulario es más sencillo y solo requiere el correo electrónico y contraseña introducidos en el registro, o los nuevos si se han modificado (la modificación de un perfil se verá más adelante).

Cerrar sesión

Si ya tiene sesión iniciada, debe hacer clic en su nombre en la barra de navegación (parte derecha). Esto abrirá un desplegable en el que, aparte de otras opciones, podrá cerrar la sesión (ver figura E.23).

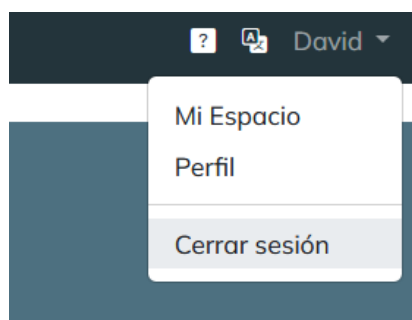


Figura E.23: Cierre de sesión

Personalizar perfil

Es posible ver el perfil propio y modificar los datos con los que se creó la cuenta.

En primer lugar, y similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Perfil» (ver figura E.24).

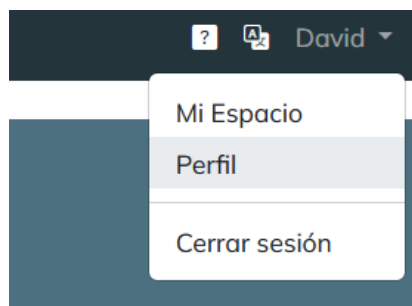


Figura E.24: Acceso al perfil

Una vez dentro, en el lateral izquierdo aparece la información general del perfil (ficheros subidos, ejecuciones, correo electrónico...).

La parte de edición (zona derecha) contiene un formulario similar al del registro (ver figura E.25). Se pueden modificar todos los datos mostrados, pero para que las modificaciones puedan realizarse, se debe introducir la contraseña actual. Si fuera errónea o no se introduce, el formulario no se enviará.

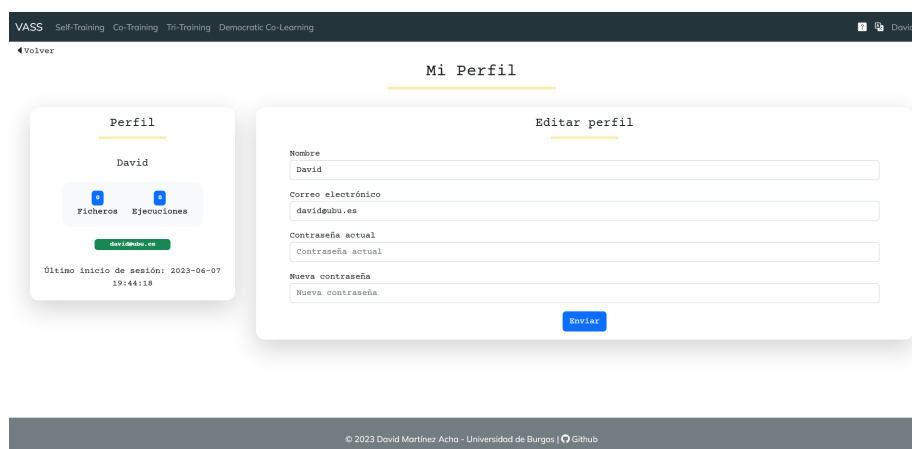


Figura E.25: Perfil personal y edición

Espacio personal

Todos los usuarios poseen de un espacio personal en el que visualizar y controlar sus ficheros subidos y las ejecuciones realizadas hasta el momento.

En primer lugar, y similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Mi Espacio» (ver figura E.26).

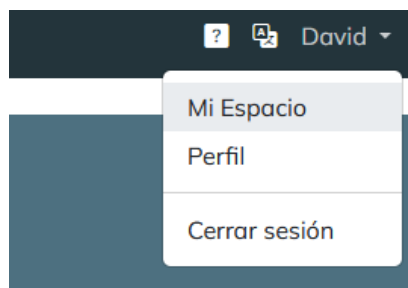


Figura E.26: Acceso al espacio personal

Una vez dentro, en el lateral izquierdo aparece la información general del perfil (ficheros subidos, ejecuciones, correo electrónico...).

En la parte derecha se encontrarán dos tablas en las que se reflejan los ficheros subidos y las ejecuciones (ver figura E.27).

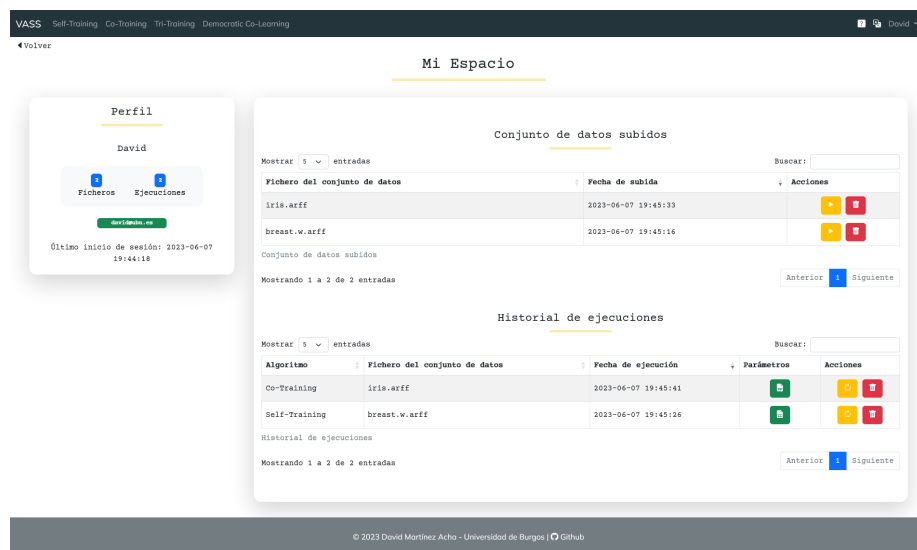


Figura E.27: Espacio personal

Ambas tablas tienen un buscador donde se puede filtrar por cualquier palabra, en todas las columnas de todas las filas. Además, puede elegir cuantas entradas mostrar (selector en la esquina superior izquierda) y en su caso, pasar las páginas para seguir mostrando más entradas (paginado en la esquina inferior derecha).

Control de los conjuntos de datos subidos Particularmente, los conjuntos de datos (ficheros) pueden ser ejecutados o eliminados.

En el caso de querer utilizar el fichero para **ejecutar un algoritmo**, simplemente se ha de pulsar en el botón con el símbolo play. Al hacerlo, se mostrará una ventana emergente (modal) para seleccionar el algoritmo deseado (ver figura E.28).

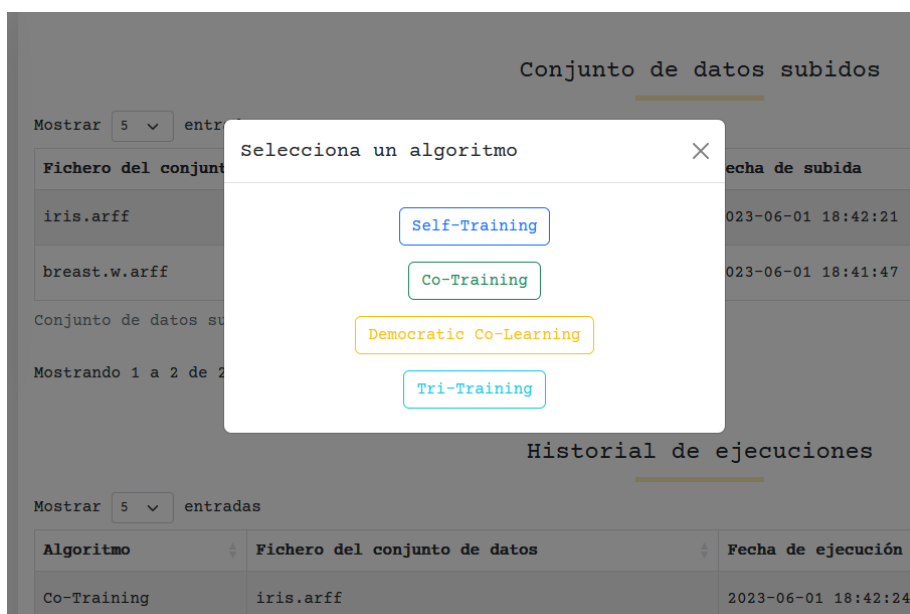


Figura E.28: Selección de algoritmo

Cuando se pulse en uno de los botones se redirige a la pestaña de configuración (ver explicación de configuración en [E.4](#)).

Por otro lado, si se quiere **eliminar un fichero** de la cuenta (y del sistema), se pulsa en el botón con el símbolo de la papelera. De igual manera, se mostrará una ventana emergente de confirmación (ver figura [E.29](#)).

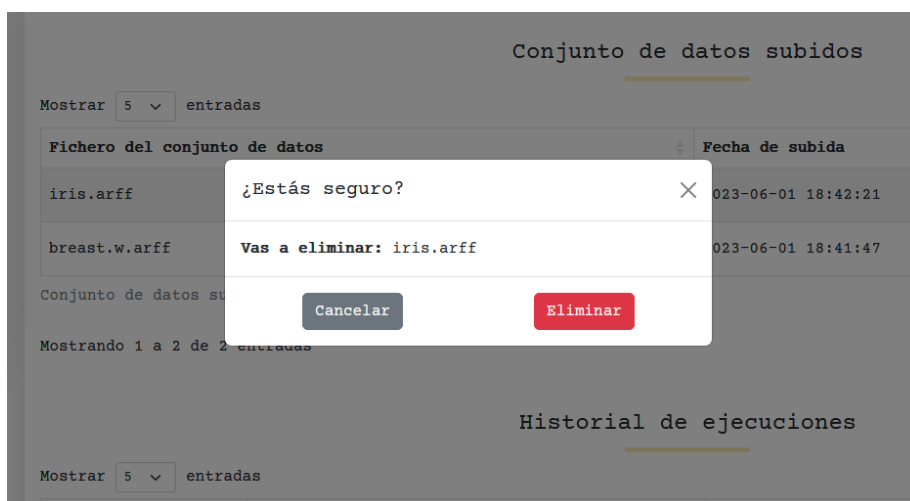


Figura E.29: Eliminar fichero

Si todo ha ido correctamente, habrá desaparecido la fila correspondiente del fichero. En caso contrario se mostrará otra ventana emergente con el error.

Control de las ejecuciones En el historial de ejecuciones también pueden realizarse varias acciones.

En primer lugar, los **parámetros de configuración** que se introdujeron en una ejecución se pueden consultar pulsando en el botón de la columna «Parámetros». Esto mostrará una ventana emergente con un JSON legible y formateado (ver figura E.30).



Figura E.30: Parámetros de una ejecución

De forma similar a los conjuntos de datos, las ejecuciones pueden ser **re-ejecutadas**, repitiendo exactamente lo mismo que ocurrió en su momento. Para ello simplemente se debe pulsar en el botón amarillo con el símbolo típico de recargar página.

Este caso no se mostrará ninguna ventana emergente, redirigirá directamente a la visualización del algoritmo (ver explicación de la visualización en E.4).

Exactamente igual a los conjuntos de datos, las ejecuciones pueden **eliminarse** pulsando en el botón con el símbolo de papelera mostrando una ventana emergente de confirmación similar a la anterior (ver figura E.31).



Figura E.31: Eliminar ejecución

E.5. Manual del administrador

Conviene dividir el manual general para usuarios anónimos y registrados de los administradores. Aun con ello, un usuario administrador puede realizar las mismas acciones que el resto de roles.

Para acceder a esta sección, se recuerda que existe un administrador público con las siguientes credenciales:

- Email: admin@admin.es
- Contraseña: 12345678

Panel de administración

El administrador puede controlar a los usuarios, todos los ficheros subidos y todas las ejecuciones. Para ello, posee de un panel de administración en el que visualizar tablas con toda esa información.

En primer lugar, para acceder al panel, similar a otros casos, en el desplegable de la barra de navegación del usuario (con rol administrador) se pulsa en «Panel de administración» (ver figura E.32).

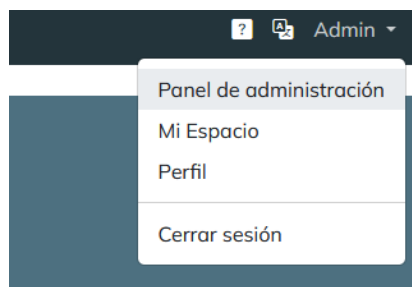


Figura E.32: Acceso al panel de administración

Una vez dentro, se tiene un menú organizado en pestañas donde ver las tres tablas (ver figuras E.33 (Administración de usuarios), E.34 (Conjuntos de datos subidos) y E.35 (Historial de ejecuciones)).

A screenshot of the 'Panel de administración' (Administration Panel) interface. The top navigation bar includes 'VASS', 'Self-Training', 'Co-Training', 'Tri-Training', and 'Democratic Co-Learning'. The main content area has three tabs: 'Administración de usuarios' (selected), 'Conjunto de datos subidos', and 'Historial de ejecuciones'. The 'Administración de usuarios' tab displays a table with columns: 'Nombre', 'Correo electrónico', 'Último inicio de sesión', and 'Acciones'. The table lists two users: 'Admin' and 'David'. A sidebar on the right shows '2 Usuarios totales'. The footer contains copyright information: '© 2023 David Martínez Acha - Universidad de Burgos | GitHub'.

Figura E.33: Administración de usuarios

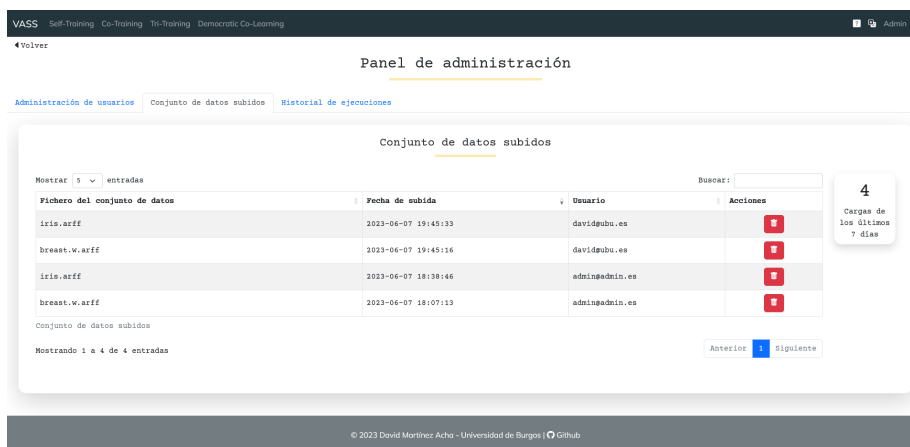


Figura E.34: Conjuntos de datos subidos

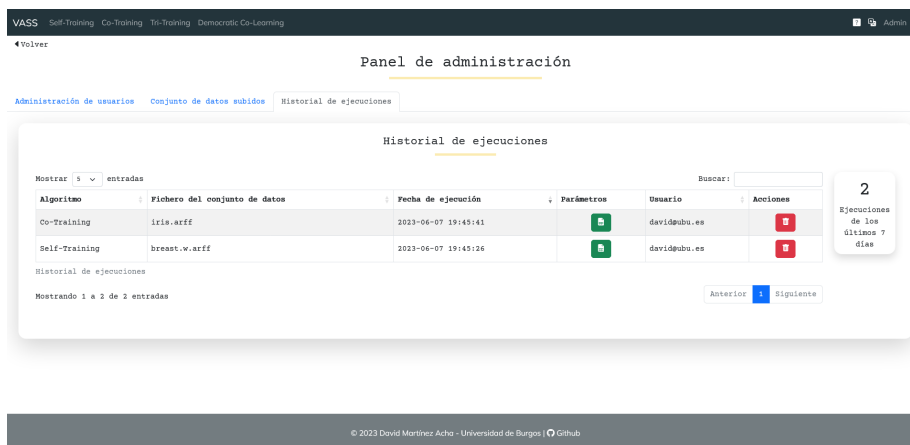


Figura E.35: Historial de ejecuciones

Para el caso de «Conjunto de datos subidos» e «Historial de ejecuciones», el proceso es el mismo que en el manual del usuario. La diferencia es que en las acciones solo puede eliminar (no ejecutar o re-ejecutar respectivamente). Consultar las explicaciones en E.4 (eliminar fichero), E.4 (mostrar parámetros de ejecución) y E.4 (eliminar ejecución).

En el caso de los usuarios, el administrador tiene dos posibilidades, editar sus datos o eliminar el usuario.

Si se quiere **editar** un usuario, se debe pulsar el botón con el símbolo del lápiz. Esto redirigirá a una pestaña similar a la de la figura E.4 (perfil del usuario) y de hecho, será como adoptar la vista del usuario que se está editando salvo por la inclusión de un indicativo como recordatorio al administrador (ver figura E.36).

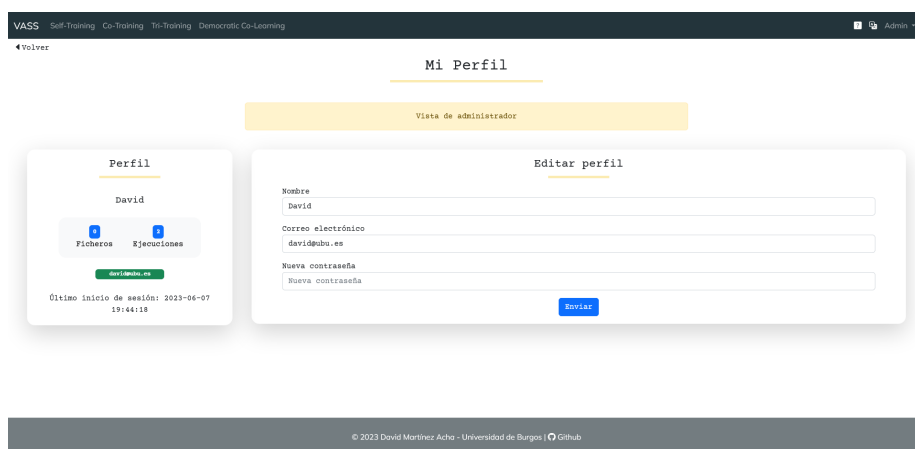


Figura E.36: Edición de un perfil ajeno

Es de destacar también que en este caso, el formulario no incluye la contraseña actual de ese usuario como confirmación. Esto es porque el administrador tiene todos los privilegios para realizar esta acción. Cuando el administrador modifique algún dato y envíe el formulario, los datos serán actualizados en el sistema.

Obviamente, los campos tienen ciertas limitaciones (similares a las del registro):

- Nombre: entre 2 y 10 caracteres.
- Correo electrónico: será el identificador del usuario en el sistema y por lo tanto solo podrá haber uno en el sistema.
- Nueva contraseña: con al menos ocho caracteres.

Si lo que se quiere es **eliminar** un usuario, el proceso es el mismo que se ha visto para el resto de eliminaciones. Se debe pulsar en el botón que incluye el símbolo de la papelera y se pedirá confirmación del proceso (ver figura E.37).

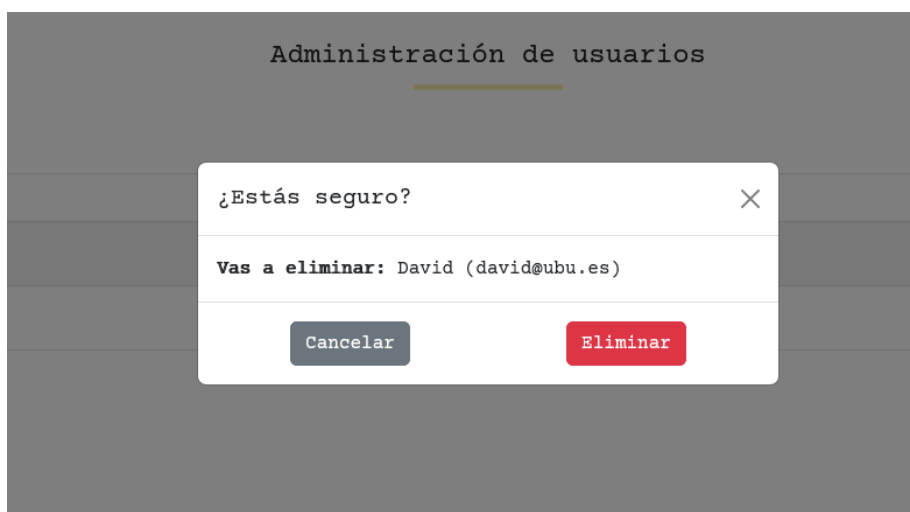


Figura E.37: Eliminación de usuario

Bibliografía

- [1] Python Software Foundation. Psf license agreement. <https://docs.python.org/3/license.html>. [Internet; accedido 26-May-2023].
- [2] José Luis Garrido-Labrador. jlgarridol/sslearn: v1.0.3.1, March 2023.
- [3] IBM. Arquitectura de tres niveles. <https://www.ibm.com/es-es/topics/three-tier-architecture>. [Internet; accedido 15-May-2023].
- [4] Seguridad Social y Migraciones Ministerio de Inclusión. Bases y tipos de cotización 2023. <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537?changeLanguage=es>. [Internet; accedido 19-May-2023].
- [5] HostGator México. Sqlite: qué es, cómo funciona y cuál es la diferencia con mysql. <https://www.hostgator.mx/blog/sqlite-que-es-y-diferencias-con-mysql/>. [Internet; accedido 15-May-2023].
- [6] OSI. The 3-clause bsd license. <https://opensource.org/license/bsd-3-clause/>. [Internet; accedido 18-May-2023].
- [7] Wikipedia. Cliente-servidor — wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Cliente-servidor&oldid=149310099>, 2023. [Internet; descargado 16-febrero-2023].
- [8] Wikipedia. Privacidad — wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Privacidad&oldid=148719517>, 2023. [Internet; descargado 20-enero-2023].

- [9] Wikipedia. Programación por capas — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Programaci%C3%B3n_por_capas&oldid=149121324, 2023. [Internet; descargado 7-febrero-2023].