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 — 16 de mayo de 2023
Tutor: Álvar Arnaiz González
Cotutor: César Ignacio García Osorio

Índice general

Índice de figuras

Índice de tablas

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 `SelfTraining` 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 `CoTraining` 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 SelfTraining y CoTraining: 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 «-1s» en el conjunto de datos para los datos no etiquetados. Así el usuario podrá subir un archivo ya *Semi-Supervisado*.
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 encontraban muchas cuestiones a la hora de resolverlo.

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 Álvar 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 Álvar 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 **??**).

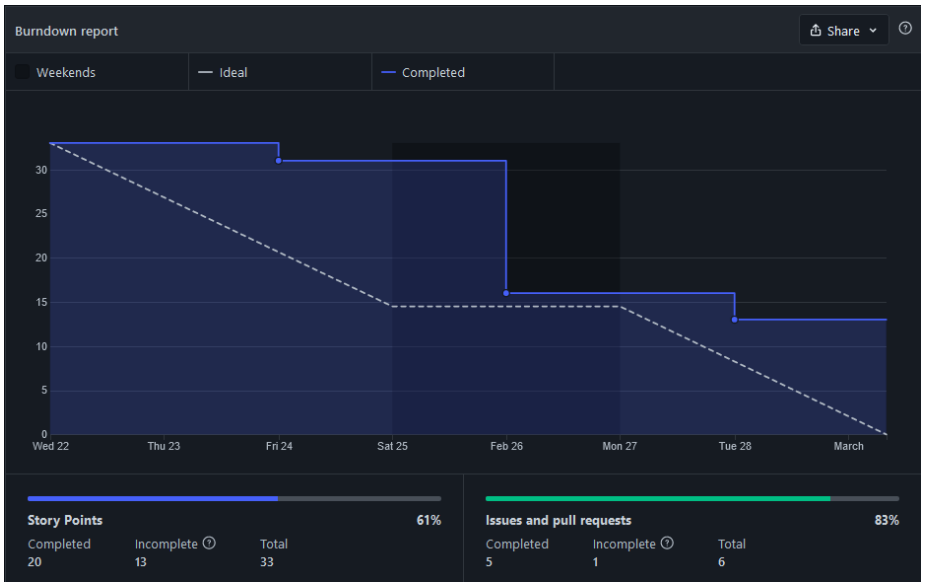


Figura A.1: Burndown chart del sprint 8.

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 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 **??**.

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.
2. Refactorización de plantillas.
3. Refactorizacion 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*, una biblioteca de José Luis Garrido-Labrador [**?**]. 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 ejemplo [1] 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 **??**.

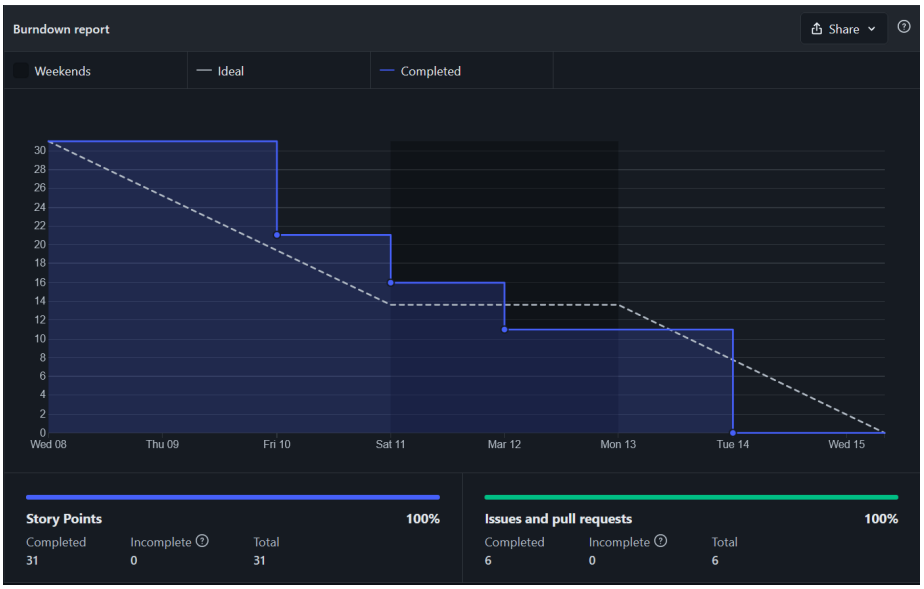


Figura A.3: Burndown chart del sprint 10.

Sprint 11

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

Puntos a desarrollar:

[1] Ejemplo 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 LaTeXy 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 **??**.

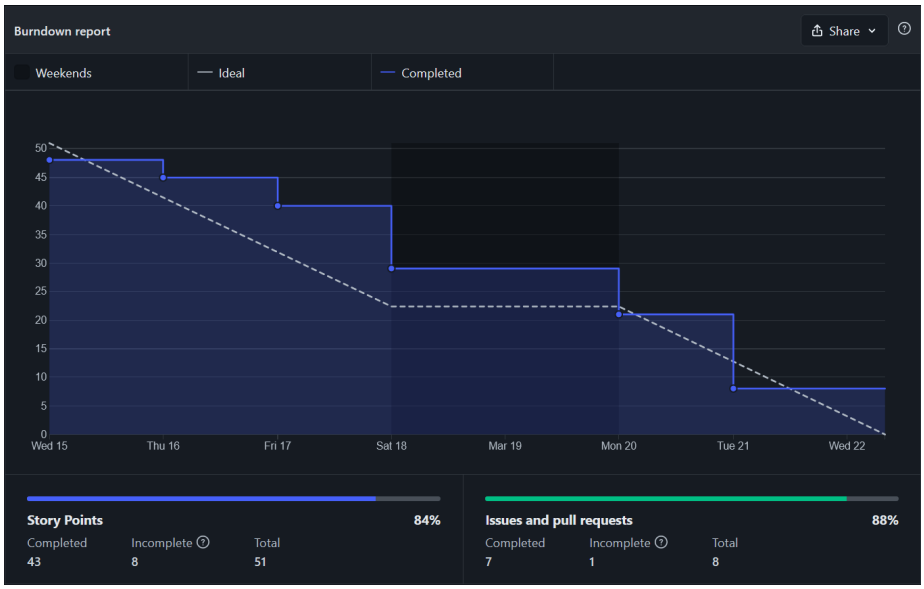


Figura A.4: Burndown chart del sprint 11.

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 cuanta 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 la 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.

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 clasificador 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 descoloran 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 documentó 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`[**?**]. 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 (1). 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. Bootstrap icons

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 conllevó 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 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:

- Añadir eliminación de ejecuciones: Al igual que usuarios y ficheros, era interesante que pudieran eliminarse las ejecuciones.
- Bug de edición perfil: 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).
- Bug reducción dimensionalidad: 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.
- 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. «Toasts» como alertas.
2. Arreglar visualizaciones (`tooltips`).
3. Cambiar formularios a WTForms.
4. Mostrar parámetros de las ejecuciones en el historial.
5. Conceptos teóricos Tri-Training.

6. 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` muestran 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 internet[2], 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 un identificador de duplicado en orden creciente.

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.

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

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

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 librería. 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 anteriomente).

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

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

- Añadir favicon.
- Control de errores.
- Arreglar bugs en las tablas del espacio personal.
- Añadir algún aspecto relevante.

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.

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 poder 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 (en principio). Además, resultaba una tarea apetecible.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se detalla la Especificación de requisitos. Aquí 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 que es lo que la aplicación debe o no 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: principal (gráfico de dos dimensiones con los puntos del conjunto de datos) y 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 deben 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 deben 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 a placer (inglés o español).

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 Privacidad**[1]: el sistema debe respetar la información privada y, de ninguna forma, compartirla con terceros. Debe ser un espacio reservado confidencial entre el usuario y la aplicación.
- **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 soportar múltiples idiomas.

B.5. Especificación de requisitos

[1]La 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 [**?**]

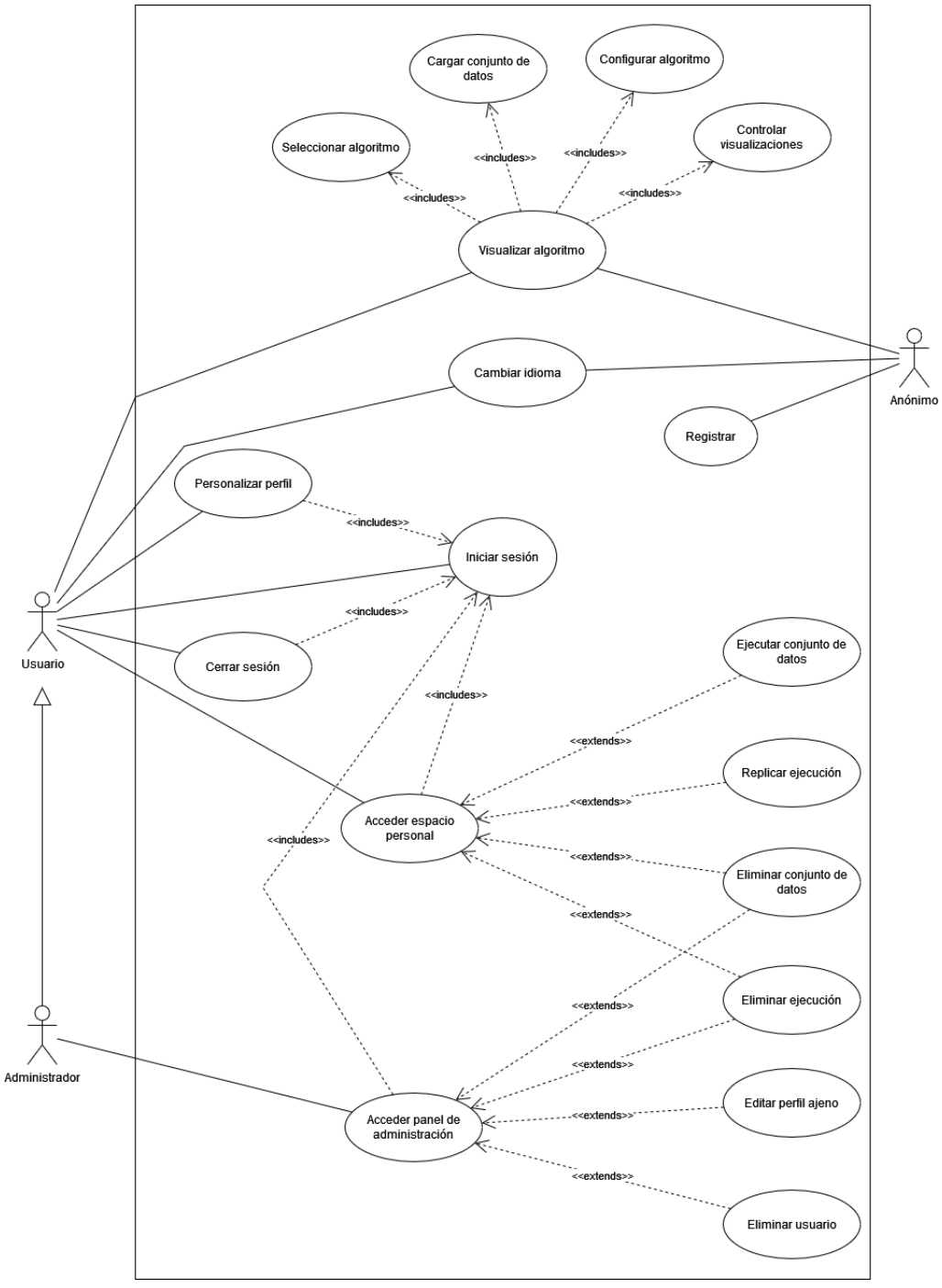


Figura B.1: Diagrama de casos de uso

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

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	1. Ejecución del Caso de Uso 2. 2. Ejecución del Caso de Uso 3. 3. Ejecución del Caso de Uso 4. 4. 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. Opcional Ejecución del Caso de Uso 5.
Extensiones	4a Si el usuario estuviera registrado, toda la información de la ejecución será almacenada.
Postcondición	Visualizaciones renderizadas en la Web
Excepciones	▪ 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 (en forma de 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: CU-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	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	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: CU-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	
	Opcional El usuario puede descargar un fichero de prueba si no lo tuviera haciendo clic en el botón de descarga. 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	
	▪ 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: CU-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	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	▪ 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: CU-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	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 *zoom* 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 los estadísticas.
Postcondición	El usuario ha podido manejar con libertad los elementos mostrados.
Excepciones	Sin excepciones
Importancia	Alta

Tabla B.5: CU-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	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 (*email*). 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	▪ Un campo del formulario no se ha rellenado. Se impedirá enviar el formulario instando al usuario a completar todos los campos. ▪ El `email` ya tiene una cuenta registrada. Se mostrará un mensaje de error debajo del campo. ▪ La contraseña de confirmación no es la misma. Se mostrará un mensaje de error debajo del campo.
Importancia	Media

Tabla B.6: CU-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	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 (*email*). 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	▪ Un campo no se ha rellenado. Se impedirá el envío del formulario instando al usuario a completar todos los campos. ▪ El `email` 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: CU-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	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: CU-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	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, *email* 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	▪ No se ha introducido contraseña actual. En este caso, se impedirá el envío del formulario instando al usuario a rellenar ese campo. ▪ 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. ▪ 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: CU-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	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: CU-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 9).
Acciones	1. El usuario decide qué conjunto de datos utilizar. 2. El usuario pulsa en el botón con el símbolo *play*. 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
Importancia	Media

Tabla B.11: CU-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 9).
Acciones	1. El usuario decide qué ejecución replicar. 2. El usuario pulsa en el botón con el símbolo de *refresh*. 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: CU-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	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 (*dashboard*). 4. Se mostrarán las tablas de los usuarios, conjuntos de datos y ejecuciones realizadas (de todos los usuarios). 5. Se mostrarán estadísticas básicas: ▪ 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 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: CU-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 9) o, si tiene rol de administrador, en el panel de administración (caso de uso 12).
Acciones	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	1. El usuario decide qué conjunto de datos eliminar. 2. El usuario pulsa en el botón rojo con el símbolo de *papelera*. 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: CU-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 9) o, si tiene rol de administrador, en el panel de administración (caso de uso 12).
Acciones	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	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 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: CU-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 12).
Acciones	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	▪ 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: CU-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 12).
Acciones	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	1. El usuario decide qué usuario 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 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: CU-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	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.
Postcondición	La página se ha cambiado al idioma seleccionado
Excepciones	Sin excepciones
Importancia	Baja

Tabla B.18: CU-18 Cambiar idioma.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se van a describir las decisiones 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

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».

Ejemplo (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 anterior 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 envuelve 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` y si no llega a ser clasificado valdrá -1.

Ejemplo (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, tres última 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 sol o 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 la etiqueta, su posición contendrá -1.
- Columna «clfs»: Hasta ahora se ha hablado de posiciones en las dos columnas anteriores. Para saber a qué clasificador se refiere esta columna contiene otra lista con los tres nombres de los clasificadores. Si

[1]La 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.

es un dato inicial, contendrá [inicio], en otro caso contendrá algo como `[CLF1(KNeighborsClassifier), CLF2(DecisionTreeClassifier), CLF3(GaussianNB)]`.

Ejemplo (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 sol o tendrá una posición y contendrá 0 (`[0]`), para el resto siempre tendrá una lista en las tres posiciones (por los tres clasificadores base). Para 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

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

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.

Ejemplo (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.

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

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.

Ejemplo:

	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

Estadísticas específicas

Este tipo de estadísticas se refiere a la necesidad de mostrar también las estadísticas particulares de los clasificadores base. Esto aplica para 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)`). Los valores 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.

Base de datos

Entidades:

- **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 (serán `One-To-Many` bidireccional).

- **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 (como se comentaba por la relación).

- **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 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 (como se comentaba por la relación).

Diagrama Entidad/Relación

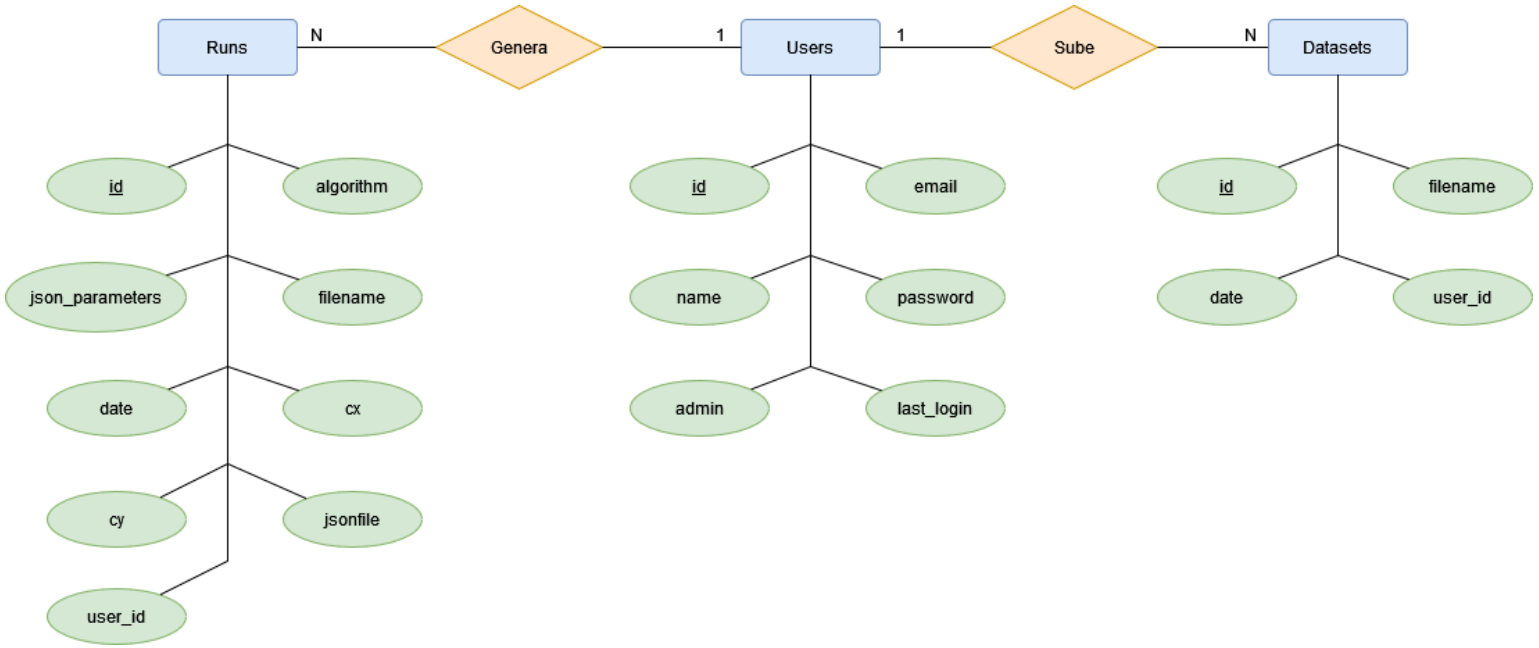


Figura C.1: Diagrama Entidad/Relación

Diagrama relacional

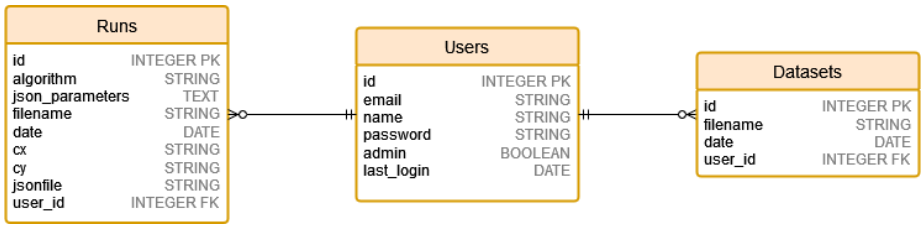


Figura C.2: Diagrama relacional

C.3. Diseño procedimental

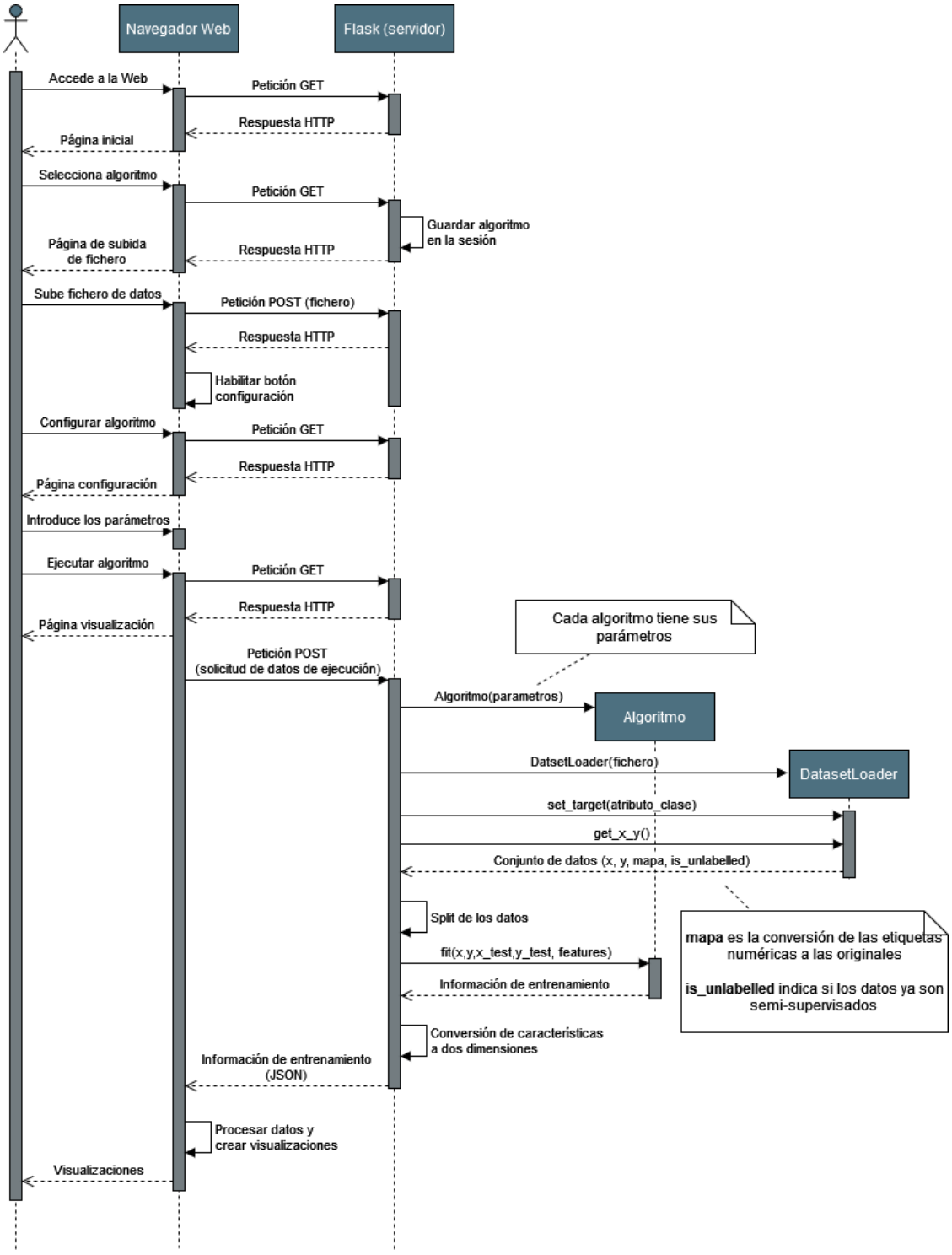


Figura C.3: Diagrama de secuencia de la visualización de un algoritmo

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) [**?**]. 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 **??** se representa un diagrama de este tipo de arquitectura.

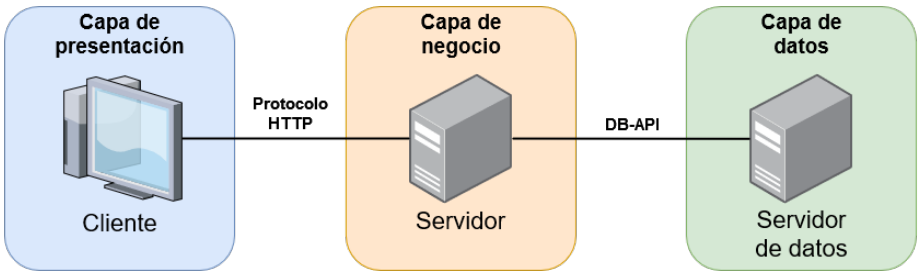


Figura C.4: Arquitectura de tres capas.

- **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 [**?**]. Se conoce también como interfaz gráfica (del sistema). Esta capa solo se comunica con la capa de negocio [**?**] (exclusivamente).

  En el proyecto, esta capa está implementada mediante todas las plantillas, estilos (CSS) y código JavaScript que con todos los formularios,

enlaces, botónes, visualizaciones... Concretamente, esto es generado por el «framework» Flask (este «framework» puede verse en este caso como la capa de presentación). Y realmente, toda esta capa se nutre de la información que solicita a la capa de negocio (por ejemplo, los datos de las ejecuciones, que no son generadas por esta capa sino por esta última).

La comunicación entre la capa anterior y la siguiente se realiza mediante el protocolo HTTP (peticiones HTTP, los conceptos pueden encontrarse en la memoria). 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 «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 correspondiente (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 [**?**] (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 **??**.

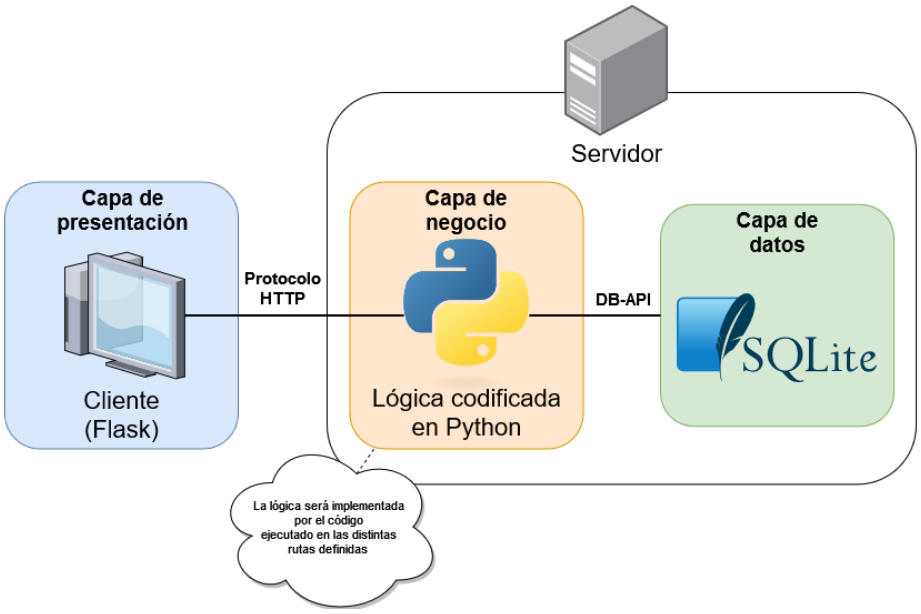


Figura C.5: Arquitectura de tres capas.

Siguen estando 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 despliegue

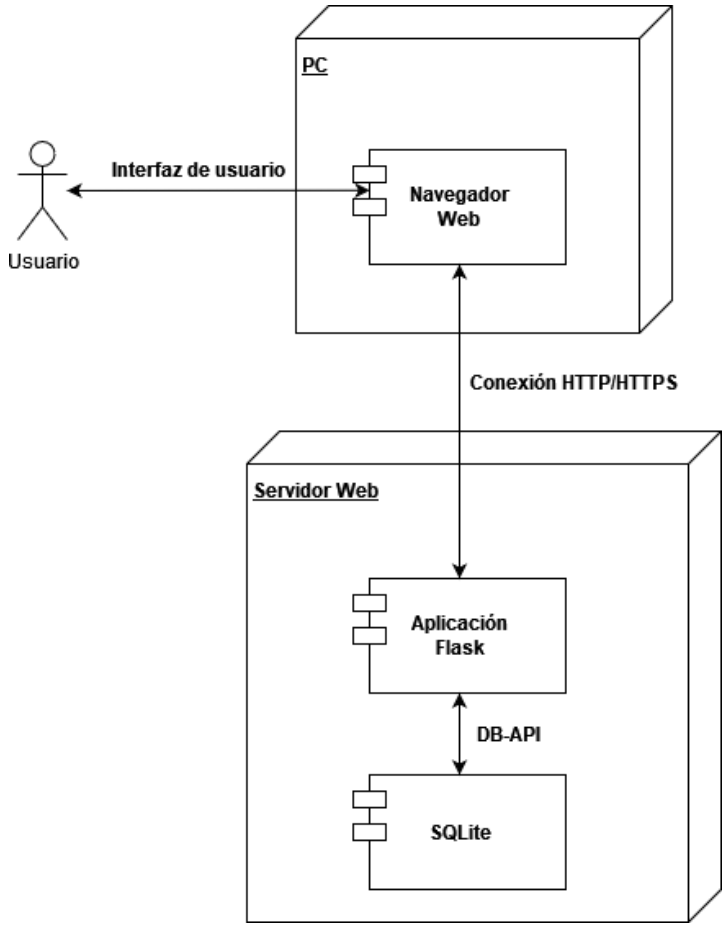


Figura C.6: Diagrama de despliegue.

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 tendrán una base común en la que aparecerá información general como la Universidad de Burgos (barra superior).

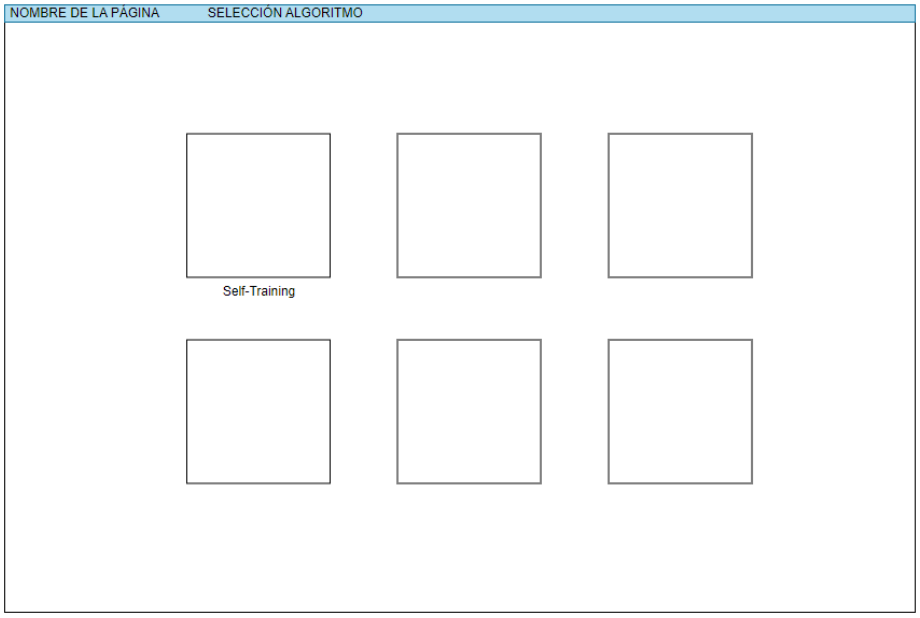


Figura C.7: Página inicial de la Web.

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.

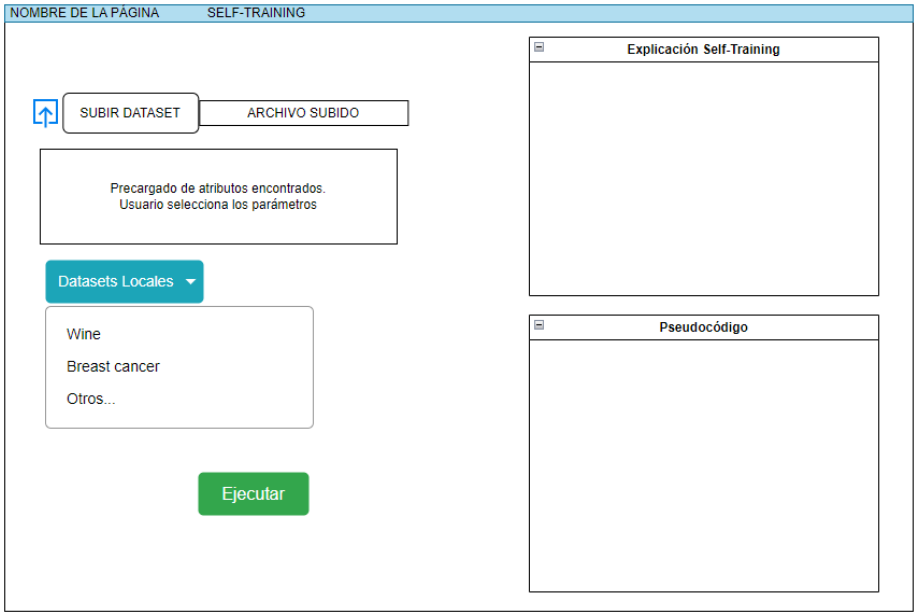


Figura C.8: Página de configuración del algoritmo.

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.

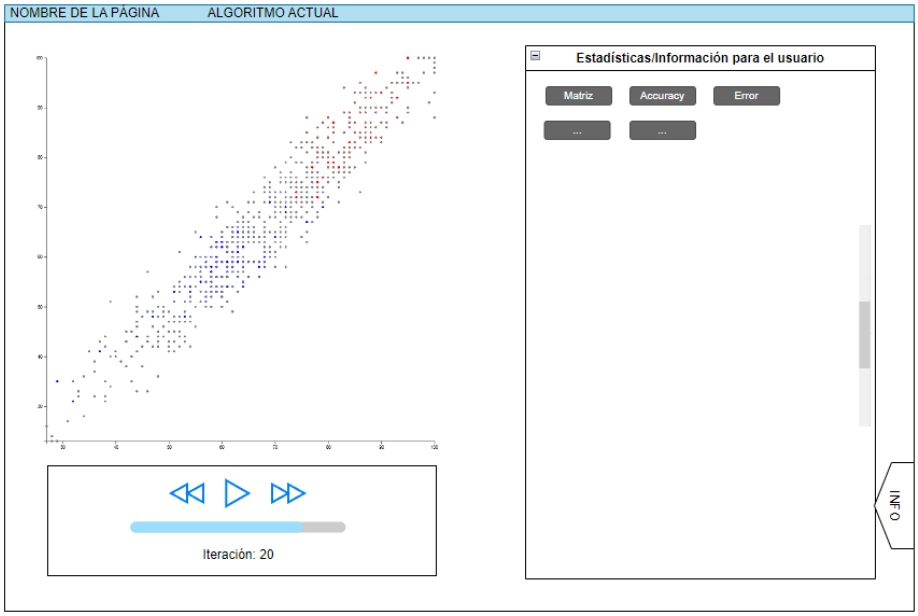


Figura C.9: Página de ejecución del algoritmo.

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.10: Barra de navegación principal

Figura C.11: Barra de navegación con usuario

Figura C.12: Animación «hover» en barra de navegación

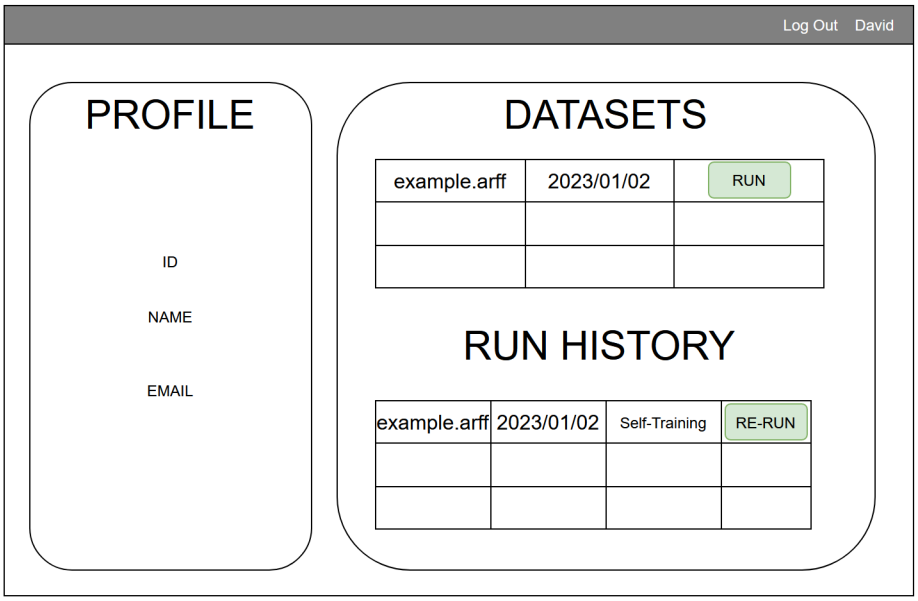


Figura C.13: Espacio personal del usuario

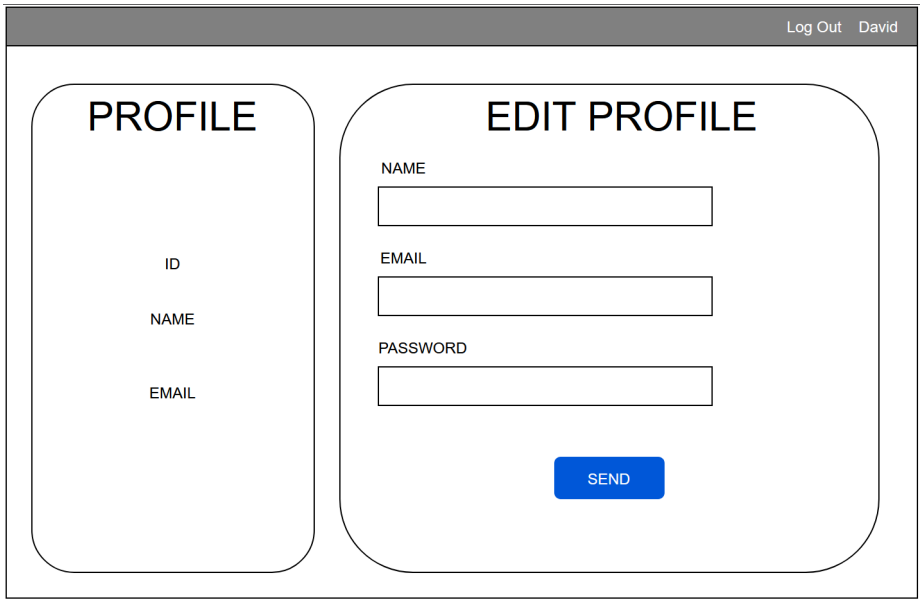


Figura C.14: visualización del perfil y modificación

Apéndice D

Documentación técnica de programación

D.1. Introducción

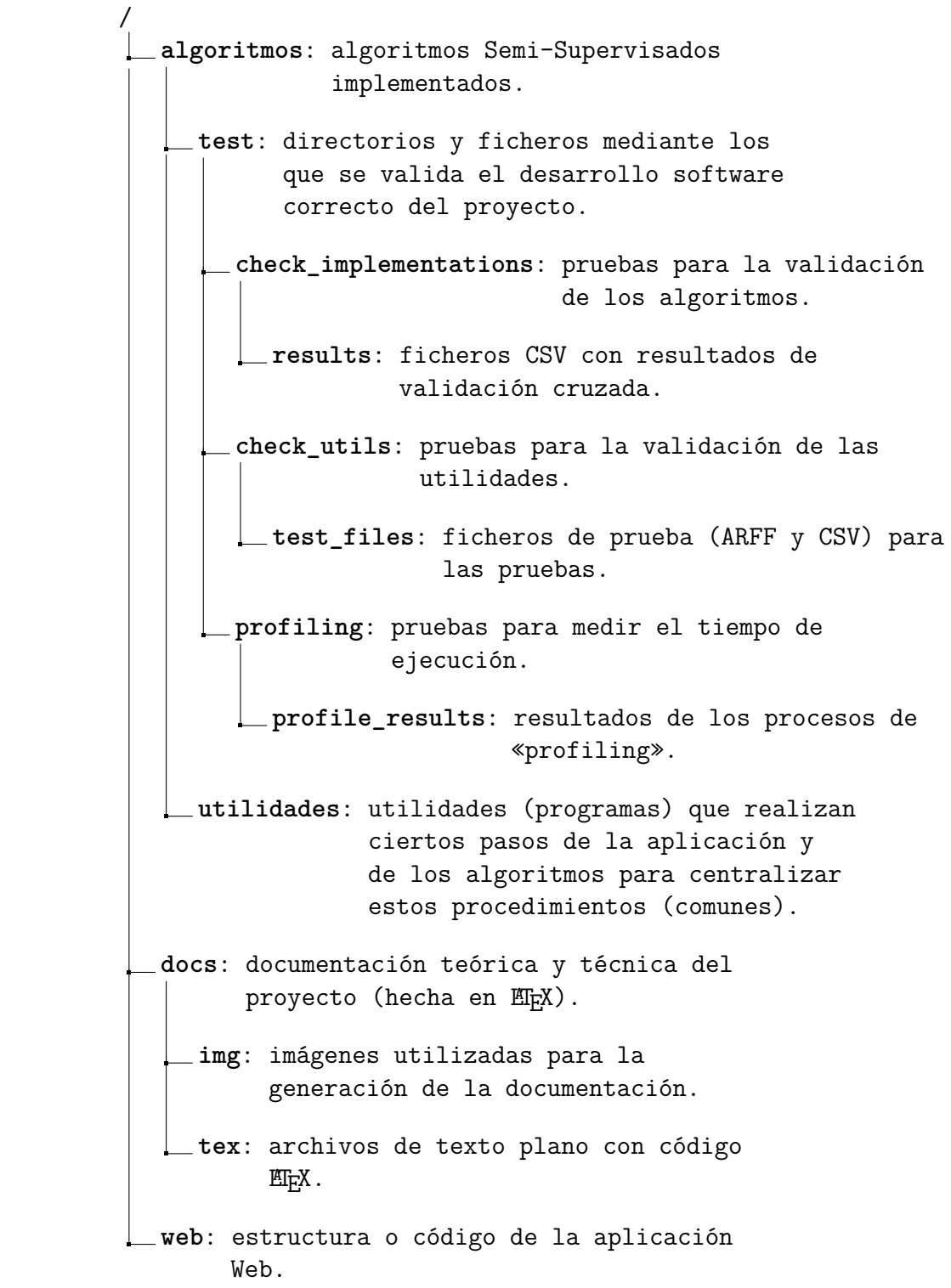
En esta sección se presenta toda la documentación técnica del desarrollo del proyecto. Trata de ser una guía para entender cómo se ha hecho el proyecto comenzando por los directorios y su contenido, con un manual introductorio para un programador que es iniciado en el proyecto, una explicación y ejemplificación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

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 (programas) que realizan ciertos pasos de la aplicación y de los algoritmos para centralizar estos procedimientos (comunes).
docs: documentación teórica y técnica del proyecto (hecha en LaTeX).
img: imágenes utilizadas para la generación de la documentación.
tex: archivos de texto plano con código LaTeX.
web: estructura o código de la aplicación Web.

- **web**: estructura o código de la 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: estructura o código de la aplicación Web.

- **app**: contiene la definición de las rutas, modelos de la base de datos y la Application Factory que instancia la aplicación.
  - **static**: ficheros estáticos que utiliza la aplicación Web: CSS, Javascript, JSON o imágenes.
    - **css**: ficheros CSS.
    - **js**: ficheros JavaScript.
    - **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).
- **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 estado utilizando para el entorno 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 principal[1]. 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].

[1] Descargas de Python: https://www.python.org/downloads/
[2] Git para Windows: https://git-scm.com/download/win

Pycharm se puede descargar tanto para Windows como Linux en la página oficial de JetBrains[3].

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, para que no haya confusión.

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 (`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)
```

[3]Pycharm: 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 instancia la aplicación, base de datos y pone 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 específica 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 necesario 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 usuario (su navegador) 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. Para incorporarlos al flujo, esto se hace incorporándolos a la selección (`/seleccionar/<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 prinicpio, 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 intervenir 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).

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.

En estos ficheros estáticos también se tienen unas imágenes con los pseudocódigos de los algoritmos. Estas imágenes deben verse tanto en la configuración como la visualización.

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 el nombre del fichero introducido.

Pero 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.

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>`». Al incluirse de esta forma, Babel si puede capturar esas funciones.

La otra posibilidad es mediante diccionarios y la declaración de una variable que indique el idioma (*locale*). 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. 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 propia 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 utilizado[4].

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

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
```

[4]Repositorio: https://github.com/dma1004/TFG-SemiSupervisado

Activación del entorno virtual

```
$ venv\activate.bat //Windows
```

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

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

```
$ python -m pip install -r requierements.txt
```

Instalar paquete de algoritmos

```
$ python -m 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
$ MD datasets\registrados
```

```
//Linux
$ mkdir runs
$ mkdir datasets/anonimos
$ 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=app.py
$ flask run
$ flask run --debug

//Linux
$ flask run
$ 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 extraidos 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 extraido los textos, se han de indicar las traducciones. 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
```

Apéndice E

Documentación de usuario

E.1. Introducción

Es 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.

E.3. Instalación

El usuario no necesita instalar el software para utilizarlo. Puede acceder a él directamente desde su navegador.

E.4. Manual del usuario

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 en el siguiente:

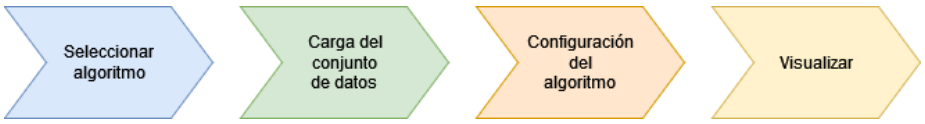


Figura E.1: Flujo de la visualización de un algoritmo

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.

IMAGEN DE LA PÁGINA PRINCIPAL INDICANDO LOS ENLACES DE LA NAV Y LAS TARJETAS.

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». 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.

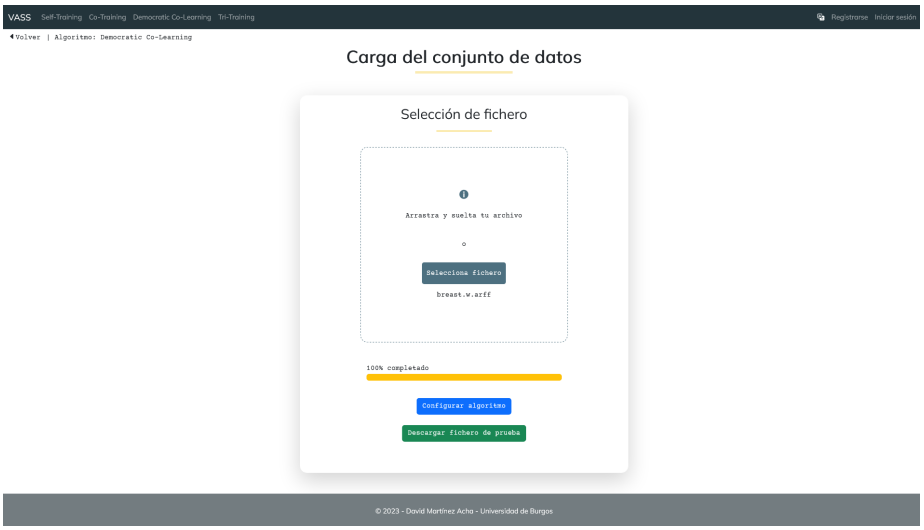


Figura E.2: Carga del conjunto de datos

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-procesamiento 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).

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.

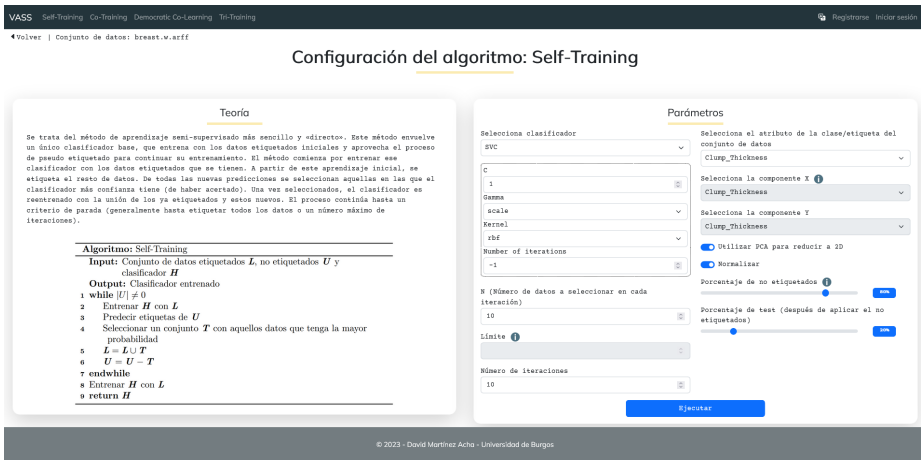


Figura E.3: Configuración del algoritmo

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, sí 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.

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.

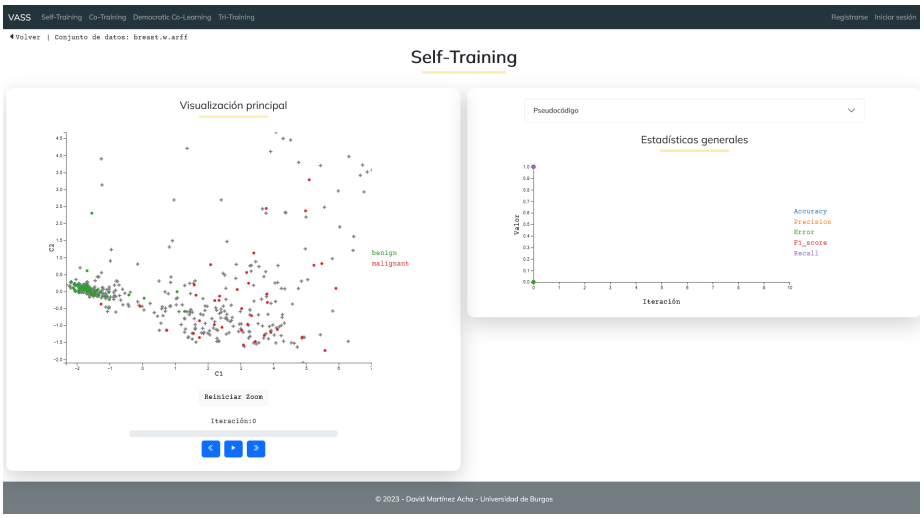


Figura E.4: Vista general de una visualización

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.

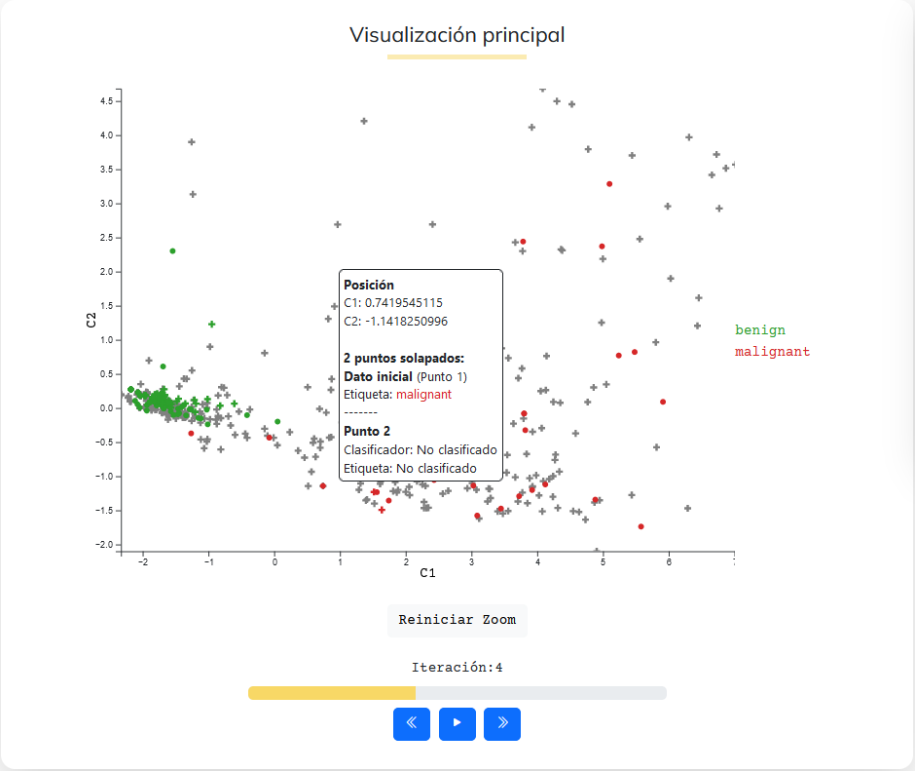


Figura E.5: Visualización principal

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.

Gráficos estadísticos En la zona de la derecha se incluirá el resto de gráficos adicionales, que serán principalmente estadísticas.

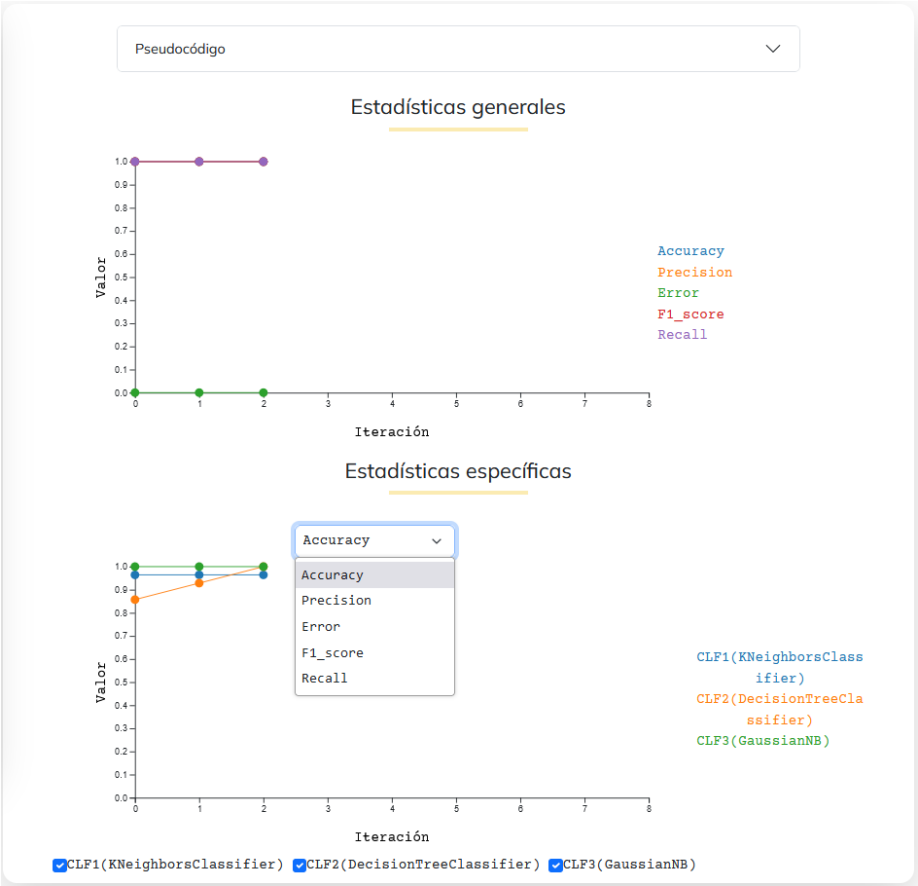


Figura E.6: Gráficos estadísticos

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:

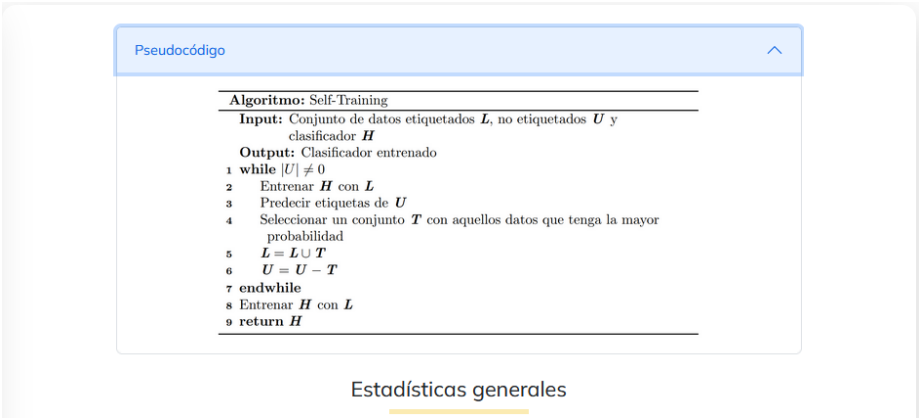


Figura E.7: Desplegable pseudocódigo

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.

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

Cambiar de idioma

Aunque la propia aplicación detecta el mejor idioma (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.

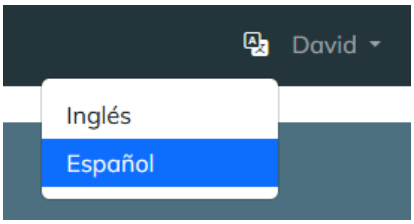


Figura E.8: Cambiar de idioma

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 (en la parte derecha) de un enlace que redirecciona al formulario de creación. 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.

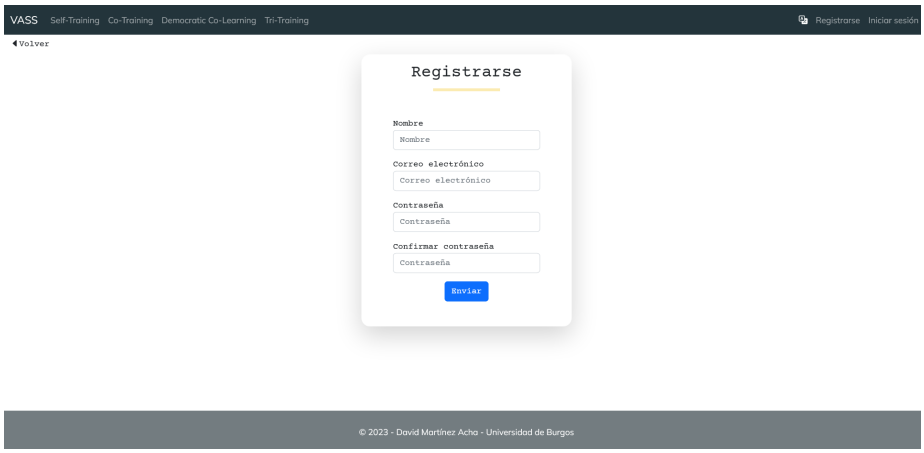


Figura E.9: Formulario de registro

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.

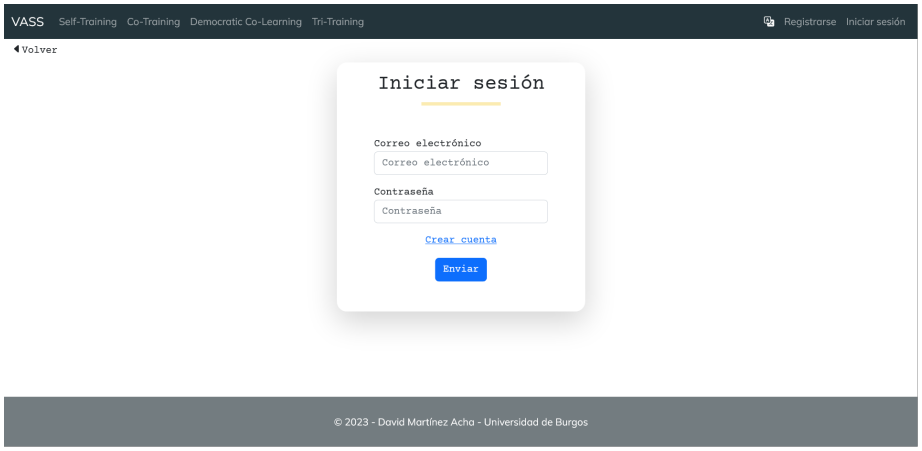


Figura E.10: Formulario de inicio de sesión

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.

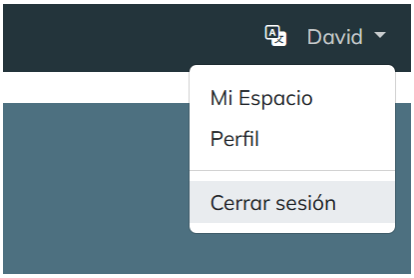


Figura E.11: Cierre de sesión

Personalizar perfil

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

En primer lugar, para acceder al perfil, similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Perfil».

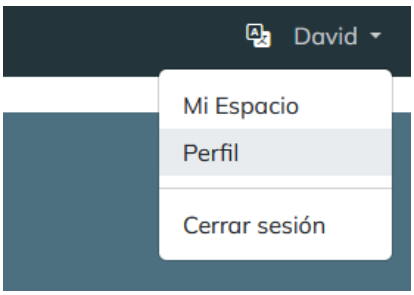


Figura E.12: Acceso al perfil

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. 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.

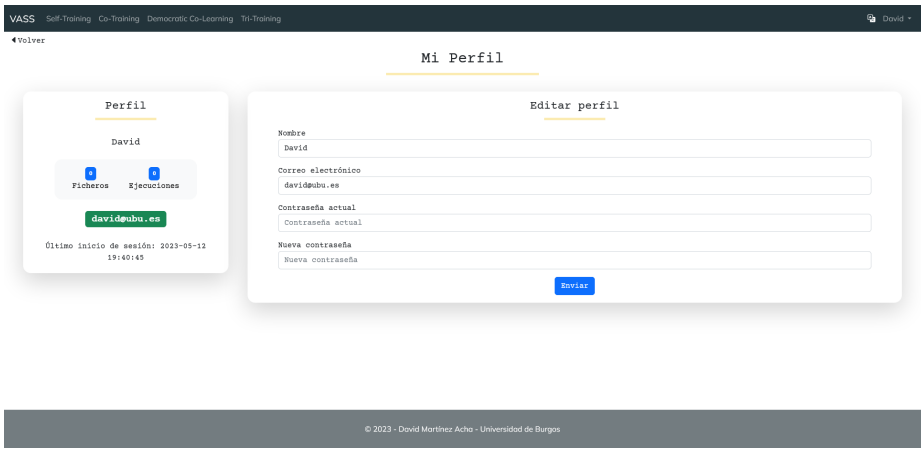


Figura E.13: Perfil personal y edición

Espacio personal

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

En primer lugar, para acceder al perfil, similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Mi Espacio».

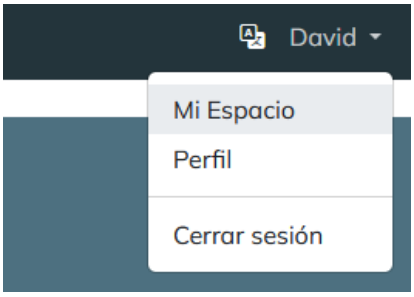


Figura E.14: Acceso al espacio personal

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.

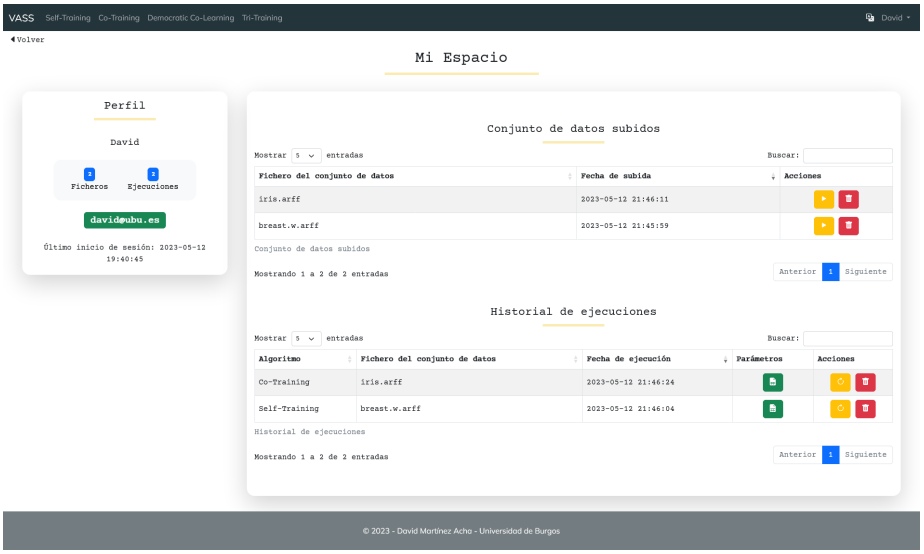


Figura E.15: Espacio personal

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 y en su caso, pasar las páginas para seguir mostrando más entradas.

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.

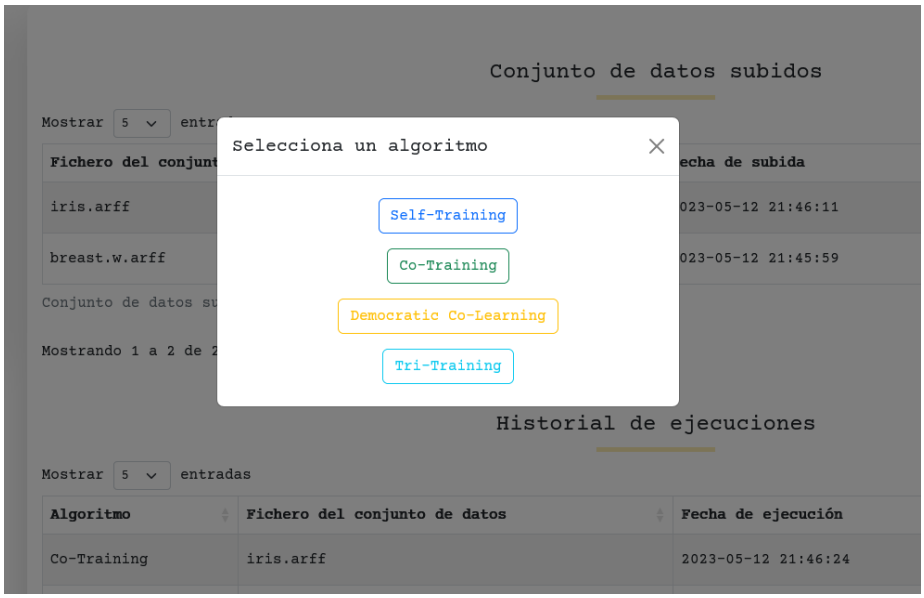


Figura E.16: Selección de algoritmo

Cuando se pulse en uno de los botones se redirige a la pestaña de configuración (**??**).

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.

Figura E.17: 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.

Figura E.18: 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.

Aunque en este caso, no se mostrará ninguna ventana emergente, redirigirá directamente a la visualización del algoritmo (**??**).

Y 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.

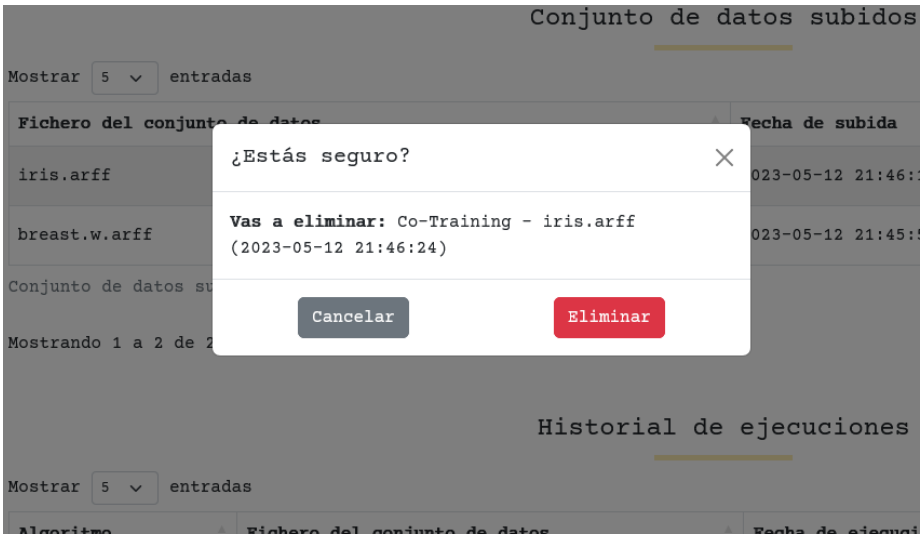


Figura E.19: Eliminar ejecución

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.

Panel de administración

Además de estas, el administrador puede controlar a los usuarios, todos los ficheros subidos y todas las ejecuciones.

El administrador 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 «Mi Espacio».

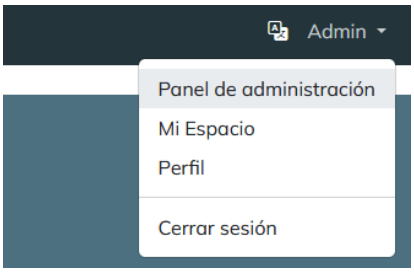


Figura E.20: Acceso al panel de administración

Una vez dentro, se tiene un menú organizado en pestañas donde ver las tres tablas.

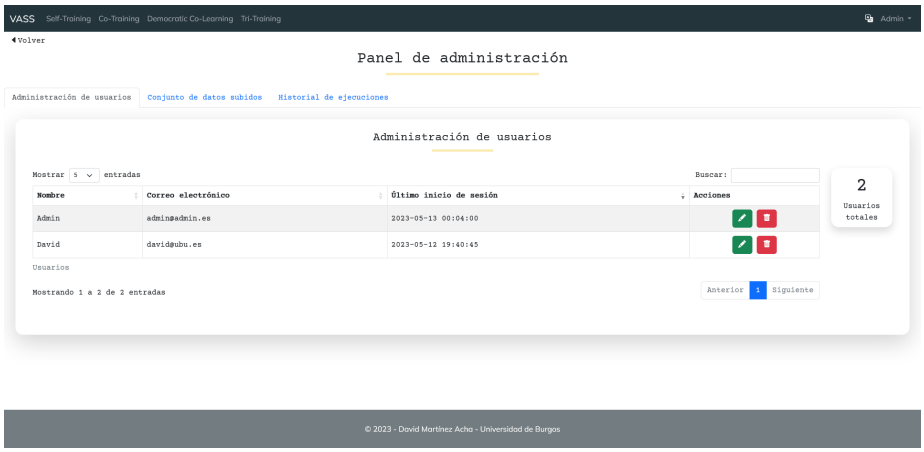


Figura E.21: Administración de usuarios

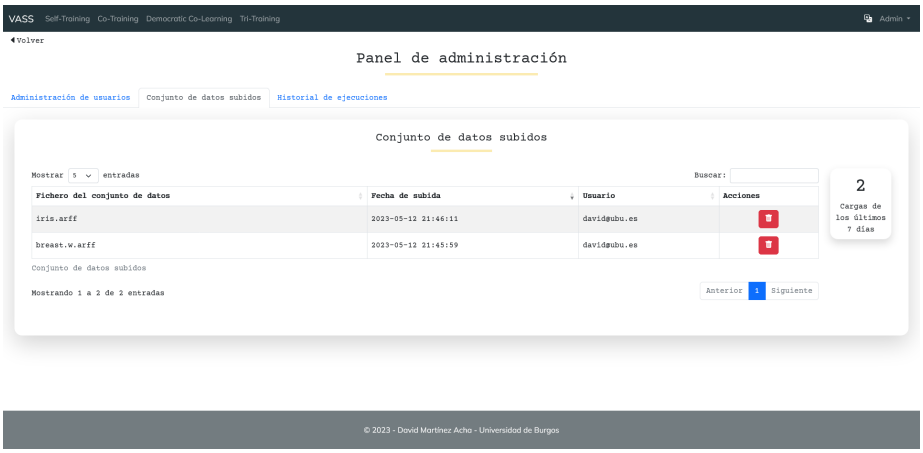


Figura E.22: Conjunto de datos subidos

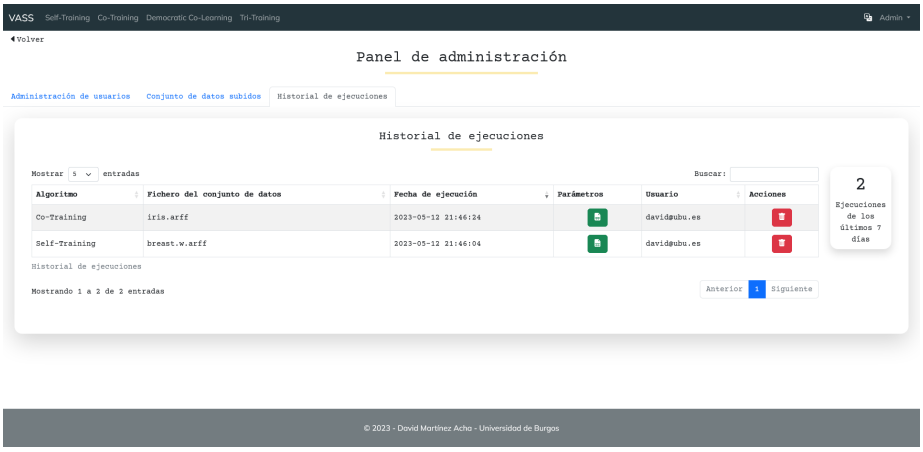


Figura E.23: Historial de ejecuciones

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

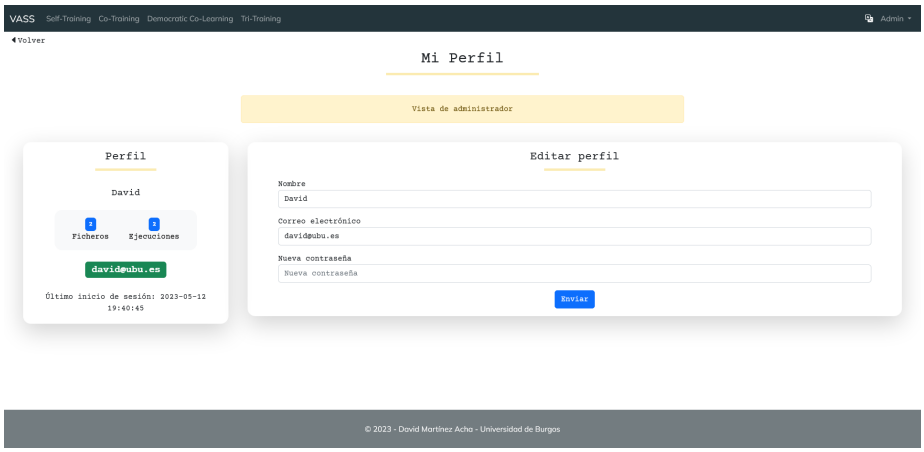


Figura E.24: Edición de un perfil ajeno

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.
- 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.

Figura E.25: Eliminación de usuario

Bibliografía

[1] José Luis Garrido-Labrador. jlgarridol/sslearn: v1.0.3.1, March 2023.

[2] IBM. Arquitectura de tres niveles. [Internet; accedido 15-May-2023].

[3] HostGator México. Sqlite: qué es, cómo funciona y cuál es la diferencia con mysql. [Internet; accedido 15-May-2023].

[4] Wikipedia. Cliente-servidor — wikipedia, la enciclopedia libre, 2023. [Internet; descargado 16-febrero-2023].

[5] Wikipedia. Privacidad — wikipedia, la enciclopedia libre, 2023. [Internet; descargado 20-enero-2023].

[6] Wikipedia. Programación por capas — wikipedia, la enciclopedia libre, 2023. [Internet; descargado 7-febrero-2023].