UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por Mario Sanz Pérez
en Universidad de Burgos — 5 de julio de 2024
Tutor: Álvar Arnaiz González

Índice general

Índice general i

Índice de figuras iii

Índice de tablas v

Apéndice A Plan de Proyecto Software 1
A.1. Introducción 1
A.2. Planificación temporal 1
A.3. Estudio de viabilidad 15

Apéndice B Especificación de Requisitos 21
B.1. Introducción 21
B.2. Objetivos generales 21
B.3. Catálogo de requisitos 21
B.4. Especificación de requisitos 25

Apéndice C Especificación de diseño 37
C.1. Introducción 37
C.2. Diseño de datos 37
C.3. Diseño procedimental 37
C.4. Diseño arquitectónico 37

Apéndice D Documentación técnica de programación 39
D.1. Introducción 39
D.2. Estructura de directorios 39
D.3. Manual del programador 42

D.4. Compilación, Instalación y Ejecución del Proyecto 45
D.5. Pruebas del sistema 49

Apéndice E Documentación de usuario **61**
E.1. Introducción 61
E.2. Requisitos de usuarios 61
E.3. Instalación 61
E.4. Manual del usuario 62
E.5. Manual del administrador 94

Apéndice F Anexo de sostenibilización curricular **99**
F.1. Introducción 99

Bibliografía **101**

Índice de figuras

B.1. Diagrama de casos de uso 26

D.1. `parametros.json` 44
D.2. Test Plan de la Web 50

E.1. Flujo de la visualización de un algoritmo 62
E.2. VASS: Página Principal 63
E.3. VASS: Carga del conjunto de datos 64
E.4. VASS: Carga del conjunto de datos con tabla 65
E.5. Dataset: Condiciones de entrada 66
E.6. Carga de datos errónea 66
E.7. Configuración del algoritmo 67
E.8. Tarjeta de teoría de grafos 68
E.9. Visualización principal 69
E.10. *Tooltip* con dato inicial 70
E.11. *Tooltip* con datos solapados 71
E.12. *Tooltip* con datos solapados con uno inicial 71
E.13. Self-Training: *Tooltip* con un dato no clasificado 72
E.14. Self-Training: *Tooltip* con un dato clasificado 72
E.15. Co-Training: *Tooltip* con un dato no clasificado 73
E.16. Co-Training: *Tooltip* con un dato clasificado 73
E.17. Co-Training: *Tooltip* con datos solapados y clasificados 74
E.18. *Tooltip* con un dato clasificado por dos clasificadores 74
E.19. Democratic Co-lerning: *Tooltip* con datos solapados y clasificados 75
E.20. *Tooltip* del Co-Forest 76
E.21. Gráficos estadísticos 77
E.22. Desplegable pseudocódigo 78
E.23. *Co-Forest*: estadísticas específicas 79

E.24. Visualización general de grafos 80
E.25. Controles de grafos 80
E.26. Visualización de un grafo 81
E.27. Nodo seleccionado en grafo 82
E.28. Instantánea de fase de inferencia en nodos 82
E.29. Grafos: datos iniciales de una clase 83
E.30. Resultados de inferencia en grafos 84
E.31. Resultados de inferencia en grafos en clase concreta 85
E.32. Cambiar de idioma 86
E.33. Tutorial 86
E.34. Formulario de Registro 87
E.35. Formulario de inicio de sesión 88
E.36. Cierre de sesión 88
E.37. Acceso al perfil 89
E.38. Perfil personal 89
E.39. Acceso al espacio personal 90
E.40. Espacio personal 90
E.41. Seleccion de algoritmo 91
E.42. Eliminar fichero 92
E.43. Parámetros de una ejecución 93
E.44. Eliminar ejecución 94
E.45. Acceso al panel de administración 95
E.46. Administración de usuarios 95
E.47. Conjuntos de datos subidos 96
E.48. Historial de ejecuciones 96
E.49. Edición de un perfil ajeno 97

Índice de tablas

A.1. Salarios brutos y costes después de impuestos 17
A.2. Costes para el *hardware* del proyecto 17
A.3. Costes para el *software* del proyecto 18
A.4. Resumen de los costes totales del proyecto 18
A.5. Librerías utilizadas, versiones y licencias 20

B.1. CU-01: Visualizar un algoritmo 28
B.2. CU-02: Seleccionar un algoritmo 29
B.3. CU-03: Seleccionar conjunto de datos 30
B.4. CU-04: Configurar un algoritmo 31
B.5. CU-05: Controlar visualización de grafos 32
B.6. CU-19: Acceder a documentación del algoritmo 33
B.7. CU-20: Ver tabla con resumen de datos de entrada 34
B.8. CU-21: Visualizar tutoriales 35

D.1. CP18: Ejecución Grafos 51
D.2. CP19: Ejecución Co-Forest 52
D.3. CP20: Subida de archivo con usuario registrado 53
D.4. CP21: Subida de archivo ilegible 54
D.5. CP22: Ejecución con fichero SS 55
D.6. CP23: Ejecución SS en grafos 56
D.7. CP24: Redirección a artículos 57
D.8. CP25: Ejecución de tutoriales 58
D.9. CP26: Ejecución desde cuenta de usuario 59

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este apéndice tiene como propósito proporcionar una visión detallada del proceso de planificación y estudio de viabilidad que ha sido fundamental en el desarrollo del proyecto de software presentado. A través de una metodología estructurada, se han abordado las diferentes tareas a lo largo del tiempo, asegurando un seguimiento exhaustivo y adaptativo de cada fase del proyecto. Además, este documento explora en profundidad el estudio de viabilidad realizado, abarcando aspectos tanto legales como económicos. El análisis legal es crucial para asegurar que el proyecto cumpla con todas las normativas y legislaciones aplicables, minimizando así posibles riesgos legales. Por otro lado, el estudio económico proporciona una evaluación detallada de la viabilidad financiera del proyecto, incluyendo estimaciones de costes y beneficios.

A.2. Planificación temporal

Como se explica en la sección 4 de la memoria, la metodología a seguir es Scrum, salvando las distancias con el número de personas que suele haber en un contexto habitual, ya que únicamente habrá un desarrollador. El proyecto se elige antes de empezar el curso, únicamente para poder informarse y leer articulos relacionados con el tema del aprendizaje semi supervisado. Al inicio con sprint más largos y a medida que se avanza en el desarrollo los sprints se reducen a una o dos semanas de duración, realizando una reunión al inicio de cada uno.

Sprint 1

Este sprint corresponde a las fechas entre el 7 de noviembre y el 18 de diciembre. Se comienza con una reunión en la que se establecen las siguientes tareas:

- Crear un repositorio en GitHub donde poder subir los cambios del proyecto, más concretamente, la plantilla de documentación inicial para ir familiarizándose con LaTeX.
- También se manda terminar de leer el artículo [12] y se asigna una nueva lectura acerca de ensembles [9]. Esto es necesario ya que de entre los algoritmos a implementar, alguno de ellos será un ensemble. Concepto que se desconocía antes de iniciar la lectura.
- Encontrar un programa adecuado para el seguimiento del proyecto que soporte Scrum.

El repositorio se crea siguiendo la plantilla de documentación ya creada en 2016 y publicada en Github, se puede acceder a través del enlace https://github.com/msp1015/TFG-Semi-Supervised-Learning. Para poder empezar a familiarizarse con LaTeX es necesario instalar los programas necesarios en el equipo local. Se prueban TeXstudio y TeXworks como editores, y por gusto y comodidad se elige TeXstudio como editor de archivos. También se instala MiKTex, que es una distribución de TeX/LaTeX para sistemas operativos Windows. Se continua con la lectura establecida, adquiriendo conceptos de ensembles, estos son, ¿Qué es el boosting? ¿Qué es el bagging? ¿Qué posibilidades hay de combinar varios modelos? entre otros.

En cuanto al programa utilizado para el seguimiento de las tareas y sprints, se prueban varios como Zenhub, Jira y Taiga. El primero era la mejor opción dado su relacion con Github, pero al ser de pago se descarta. Entre Jira y Taiga se elige la segunda por su accesibilidad, los numerosos problemas de Jira hacen que la plataforma de Taiga sea la elegida. Esta herramienta se explica en el apartado cuatro de la memoria, pero su sencillez y el hecho de poder comunicarse con GitHub hacen facil su uso. Aun asi, queda abierto a cambiar debido a que no es la herramienta que ofrece mas posibilidades.

Sprint 2

Sprint correspondiente a las fechas entre el 18 de noviembre y el 15 de enero. Se inicia con una reunión previa para establecer las tareas:

- Finalizar la lectura de [9].
- Comenzar la documentación con conceptos teóricos acerca del aprendizaje automático vistos hasta el momento.
- Aprender a usar el entorno de flask en python.

La lectura y documentacion se realiza durante el periodo de vacaciones, mientras que el aprendizaje de flask, se lleva a cabo con la ayuda de la asignatura cursada de Diseño y Mantenimiento del Software, en la que se realiza una web que se implementa con este *framework*.

Se encuentra una aplicación llamada Zappier, la cual permite construir triggers entre aplicaciones. En este caso sirve para que cada vez que se cree una *issue* en github, se cree como historia de usuario en el product backlog de Taiga, donde despues se gestionará independientemente. Esta aplicación está de nuevo explicada en el apartado 4 de la memoria.

Sprint 3

Sprint correspondiente a las fechas entre el 15 de enero y el 1 de febrero. Se tiene una reunión previa para asignar las tareas de:

- Lectura del algoritmo Co-Forest [7] y su pseudocódigo.
- Búsqueda de trabajos relacionados para coger ideas propias para el proyecto.
- Continuar documentando los conceptos teóricos (ensembles, Co-Forest).

En este periodo se completa la lectura del artículo del algoritmo Co-Forest y del apartado del trabajo de Patricia y sus estudios relacionados con este algoritmo. El principal estudio que realiza consiste en resolver un error del pseudocódigo, donde un valor podía coger el valor 0 cuando se utiliza como divisor. Mediante tres propuestas, se inicializa este valor con diferentes operaciones y se muestran varias gráficas para poder evaluar la mejor opción.

Se realiza una búsqueda amplia de aplicaciones web para visualización de algoritmos, apuntando y explicando las más interesantes en el apartado 6 de la memoria. Se realiza la implementación de la técnica de Scrum en el apartado 4 de la memoria. Se deja para sprint posteriores la documentación del CoForest, ya que puede que los conceptos adquiridos no sean los correctos hasta que no se haga su implementación y se vean los resultados.

Sprint 4

Sprint correspondiente a las fechas entre el 1 de febrero y el 14 de febrero. Las tareas asignadas para este sprint son:

- Primera implementacion del algoritmo Co-Forest, basado en [7].
- Evaluar esta primera implementación.
- Actualizar documentación, incluyendo este apartado.
- Continuar con la búsqueda de trabajos relacionados en la web.

Para la primera tarea, es importante comentar que se utiliza como referencia el pseudocódigo del articulo mencionado pero también la implementación de Patricia Hernando [4]. Una vez se tiene la primera implementación del algoritmo, se compara con el algoritmo de Patricia, el cual se considera una solución muy buena como ensemble. El estudio realizado se explica mejor en el apartado de aspectos relevantes de la memoria, pero cabe mencionar que los primeros resultados no son muy fiables, lo que hace pensar que algo está mal implementado. Además de la búsqueda de páginas anteriores, se encuentra una muy buena opción: `https://ml-visualizer.herokuapp.com/`. Esta página tiene algo diferente, ya que permite configurar y ver el resultado del algoritmo en la misma ventana. Este será uno de los objetivos en la web.

Sprint 5

Sprint que corresponde a las dos semanas siguientes, se asignan las tareas:

- Corregir fallos en la implementación del Co-Forest.
- Comenzar a mirar el código correspondiente a la web del trabajo de David, el cual podría resultar interesante para el futuro.
- Evaluar correctamente el Co-Forest.

- Continuar con la documentación.

Se muestra al tutor la página encontrada, se acuerda que puede ser una buena idea para la web, pero antes hay que familiarizarse con el entorno de la web y su código. Esto incluye decidir si las llamadas a la web, una vez ejecutas el algoritmo, se realizan de una en una, devolviendo un gráfico en cada iteración, o una sola llamada que calcule todo el algoritmo y que reciba un diccionario con toda la información necesaria para su representación. Se resuelven dudas del algoritmo Coforest, como la necesidad del uso de *random state* para tener un estudio reproducible. El estudio sigue teniendo bastantes diferencias en comparación con el de Patricia, lo que hace pensar de nuevo que algo en el código no esta bien programado.

Sprint 6

Se inicia con una reunión el 7 de marzo, surge un cambio de planes, dejando las siguientes tareas por hacer:

- Continuar documentación (Co-Forest, trabajos relacionados)
- El trabajo será una versión 2.0 del desarrollo de David.
- Esto implica tener que familiarizarse con el trabajo de David, su estructura, web, etc.
- Corregir estilo de programación. Estandarizar mediante la guía de estilo de python, PEP8 [13].
- Revisar Co-Forest debido a que la comparación con el de Patricia no es del todo buena.

La decisión de continuar el trabajo de David se da ya que la idea original para esta implementación iba a ser prácticamente igual. Por ello, se aprovecha la base de este trabajo, sobretodo el de la web, ya que los algoritmos serán diferentes. La idea es seguir con un estilo propio en la configuración de la web, pero todo sobre la plantilla ya creada por David. En cuanto al código implementado hasta el momento, se considera que está bastante desordenado, sin documentar y sin seguir una guía de estilos. Por ello, se establece el idioma español para nombrar a todas las variables, se documentan todos los métodos de la manera correspondiente en python, y con la auyda de librerias como *pylint* y *mypy* se sigue la guía de estilo PEP8. El error que se estaba cometiendo en cuanto a la implementación del Co-Forest estaba en

el calculo del error, ya que uno de los parámetros (los índices de los datos de entrenamiento que se usan en cada árbol) se estaba repitiendo en cada ejecución, cuando cada árbol debería tener los suyos. Es decir, el error se estaba calculando de manera incorrecta. Una vez corregido esto, el algoritmo se considera eficiente.

Sprint 7

Este Sprint es aún más reducido debido a las condiciones dadas, correspondiente entre los días 14 y 20 de marzo. Se utiliza para avanzar en todas aquellas tareas retrasadas, asignando las tareas:

- Documentación de conceptos teóricos: tanto teoría de ensembles como el propio algoritmo Co-Forest.
- Documentación de trabajos relacionados.
- Documentación de aspectos relevantes.
- Documentación de anexos.
- Prototipo de ventana pensada para la configuración del algoritmo.

Sin mucha más explicación, se avanza todo lo que se puede en la documentación y, una vez visto y entendido la mayor parte del proyecto web de David, se empieza el prototipo de una nueva ventana.

Sprint 8

Debido a que el calendario corresponde con la semana santa, este Sprint corresponde entre los días 20 de marzo y 4 de abril. Se asignan las siguientes tareas:

- Continuación de toda la documentación del Sprint anterior.
- Ver tutoriales de JavaScript y de BootStrap
- Introducir correctamente el Co-Forest en la web
- Investigar la manera en la que se pasan los parametros de ejecución del algoritmo (JSON).

En cuanto a los tutoriales, se siguen los de la web https://www.w3schools.com/ y también algún video de YouTube. HTML y css son dos lenguajes que se pueden ir aprendiendo sobre la marcha, pero javascript puede ser complicado de entender sin unos conceptos previos, y más cuando se trata de un proyecto complejo. Por esto se dedica gran parte del tiempo a aprender las bases del lenguaje. En cuanto a la web, se integra gran parte del Co-Forest, permitiendo realizar todos los pasos hasta la visualización, donde surge un error de servidor. Para conseguir la comunicación entre la ejecución del algoritmo y su visualización se realizará siguiendo la misma técnica que David, para ello primero hay que comprender todo lo que recoge del propio algoritmo. Se decide fijarse en el algoritmo Democratic Co-Learning por su gran parecido con el Co-Forest.

Además de todo esto, aparece un error en la parte de la gestión de usuarios, no permitiendo registrar ni logear usuarios. Esto se debe a algún problema con el método que se utiliza para la encriptación de las contraseñas, el cual se deja a resolver para el siguiente sprint.

Sprint 9

De nuevo se vuelven con las reuniones semanales, estableciendo las siguientes tareas:

- Introducir la parte de visualización del co-forest en la web.
- Modificar el código del Co-Forest para tener el JSON.

Ambas tareas están relacionadas, ya que para poder ver una visualizacion final, es necesario almacenar todos los datos de la ejecución. La tarea de conseguir la visualizacion consiste más bien en corregir el error mencionado anteriormente. Ya que un error 400 no da muchas pistas de donde puede estar el error, se hace un seguimiento de todo el proceso. Esto conlleva ir mostrando por consola los diferentes valores y resultados. Finalmente se localiza el error en la forma en la que se denominan los *div* en los formularios. Para el caso del Co-Forest, se usa lo que ya existía de los árboles de decisión. Esto hace que no haya necesidad de crear un *div* para seleccionador el clasificador. Sin embargo, para aprovechar el código de David, es necesario definir uno y darle el mismo nombre en los diferentes formularios para que funcione. El JSON se consigue basandose de nuevo en el Democratic Co-Learning. Puede que se guarden los mismos datos, pero la manera de recogerlos es distinta en cada algoritmo.

Se resuelve el error del sprint anterior de la parte de gestión de usuarios. Una vez aislado el error se ve que lo que falla es un método de una librería utilizada para encriptar contraseñas, lo que da a pensar que alguno de los parámetros pasados deja de ser compatible con la versión de la biblioteca actual. Efectivamente la versión de este proyecto de la biblioteca *werkzeug* no coincidía con la del trabajo base. En concreto el parámetro incompatible que se le estaba pasando a este método es la propia contraseña encriptada, la cual empezaba por *sha256*. La nueva versión del método establece que para usar *sha256* el texto debe empezar por *pbkdf2:sha256*, por lo tanto se cambia esta opción y también el *hash* definido para el administrador al inciar la aplicación.

Sprint 10

Del 11 de abril al 18 de abril, se establecen las siguientes tareas de cara al siguiente sprint:

- Actualizar documentación.
- Visualizar resultados del Co-Forest en la web
- Crear un gráfico de tarta como tooltip en los datos.

Se consigue la visualización del Co-Forest completa reutilizando gran parte del código de los otros algoritmos, dando pie también a posibles cambios futuros en la visualización. Finalmente se determina que el gráfico de tarta no es la mejor manera de representar este tipo de soluciones y se mantiene el estilo original. Esto tiene la desventaja de que en el Co-Forest existen muchos más clasificadores y en ocasiones provoca un *tooltip* mucho más grande y que se sale de la pantalla.

Además se hacen pequeños cambios, por ejemplo se considera que es mejor dejar una opción de la etiqueta o clase por defecto al configurar cualquier algoritmo. Esto es así porque cuando tratamos con archivos dedicados a aprendizaje automático, la mayoría de ellos tienen su etiqueta en la última columna. Por lo tanto se mostrará por defecto esta columna como clase o etiqueta en vez de dejarlo vacío. También se añade una opción que permite desmarcar y marcar todas las opciones (clasificadores) en el gráfico de estadísticas específicas, esto surge como idea después de implementar el Co-Forest, ya que si el usuario selecciona muchos árboles de decisión, está bien que pueda quitar o añadir todos de golpe.

Sprint 11

Sprint correspondiente a los días entre el 18 y el 25 de abril, en la reunión se establecen las siguientes tareas:

- Finalizar tareas pendiente del Co-Forest. Estas son:
  - Arreglar última iteración ya que no aporta información
  - Quitar parámetro de inicialización de pesos ya que realmente no afecta demasiado.
  - Cambiar la visualización del *tooltip* para que se pueda ver toda la información.
- Leer documentación de construcción de grafos.
- Elegir biblioteca de *python* para la construcción de grafos.

Este Sprint se utiliza sobretodo para leer e investigar acerca de los métodos transductivos o basados en grafos. Más concretamente se manda leer el artículo del algoritmo *GBILI*, que será uno de ellos a implementar en cuánto a la construcción del grafo y el artículo de *Local and Global Consistency (LGC)* como algoritmo de inferencia de etiquetas. En la memoria se especifican las características de ambos algoritmos.

Se realiza un estudio para la elección de una biblioteca en el uso de grafos, que se puede encontrar en los aspectos relevantes. Finalmente se decide utilizar *NetworkX* por su facilidad de uso y aprendizaje.

En cuánto a los arreglos del Co-Forest, se realizan las dos primeras tareas marcadas y se deja para más adelante cambiar la visualización del *tooltip*. El problema con la última iteración provenía de la manera en la que se devolvía el número de iteraciones. Los datos se guardaban correctamente, pero teniendo en cuenta que las iteraciones comienzan en 0, se estaba devolviendo un valor mayor del que debería.

Sprint 12

Sprint correspondiente a la siguiente semana entre el 25 de abril y el 2 de mayo. Se mandan las siguientes tareas:

- Implementar algoritmos: tanto el *GBILI* como el *LGC*.

- Documentar los conceptos de cada algoritmo y los aspectos relevantes en su implementación.

En este sprint no hay mucho detalle que comentar ya que se trata de implementar el algoritmo. Finalmente no se utiliza la librería *NetworkX* para almacenar los datos del grafo, sino que se utiliza para su visualización. La visualización en este caso ayuda a ver si realmente el grafo se está construyendo siguiendo los pasos establecidos y si tienen sentido o no. Por ejemplo, si se colorean todos los nodos que son etiquetados, se podrá saber si la conexión basada en estos nodos es correcta o no.

Finalmente no se consigue implementar ambos algoritmos de forma definitiva pero sí se llega a una buena aproximación. El aspecto a destacar es que de la forma en que se implementa el algoritmo *LGC* realmente no aprovecha la disposición física final del grafo, sino su matriz de distancias original. Se va con esta premisa a la siguiente reunión para tener en cuenta un posible cambio.

Sprint 13

Corresponde a los días entre el 2 y el 9 de mayo. Se comentan las siguientes tareas a realizar:

- Depurar el código del algoritmo *GBILI*.
- Depurar el código del algoritmo *LGC*.

Se trata de finalizar ambos algoritmos para poder empezar su integración con la web.

El resumen de las correciones en el *GBILI* (todas ellas comentadas en la sección 5 de la memoria) es: el grafo es no dirigido por lo que se deben almacenar enlaces en ambas direcciones, el pseudocódigo se puede interpretar erróneamente y provocar fallos al almacenar las estructuras de datos necesarias y por último, la visualización debe ser con el grafo ordenado para que tenga sentido las nuevas conexiones.

En cuanto a la inferencia se debe tener en cuenta la conexión entre nodos del grafo mediante una matriz de afinidad binaria (1 si hay conexión, 0 en caso contrario). Se resuelve así el problema de que el algoritmo no utilizaba los enlaces del grafo, pero surge uno nuevo. Uno de los pasos de este algoritmo hace la suma por filas de esta nueva matriz de afinidad,

consiguiendo así valores enteros en su diagonal. La posibilidad de que haya ceros en esta matriz hace que se produzcan indeterminaciones en operaciones posteriores, por lo que se debe plantear una solución a esto.

Se consigue tener el algoritmo *GBILI* bien implementado, pero al inferir etiquetas los resultados no son los esperados con la nueva matriz de afinidad.

Se empieza a integrar en la web una nueva opción para seleccionar la visualización de grafos. La idea es tener una única opción y combinar la construcción del grafo junto con la fase de inferencia de etiquetas. A estas alturas se planea hacer dos algoritmos de construcción de grafos y dos de inferencia (ya teniendo uno de cada).

Sprint 14

Correspondiente a los días entre el 9 y el 16 de mayo. Se comentan las siguientes tareas:

- Corregir definitivamente ambos algoritmos (*LGC* en especial).
- Integrar en la web, con la visualización por fases.

En la reunión se dan las pautas finales para implementar definitivamente el algoritmo *LGC*. Se estaba utilizando la matriz de afinidad definida por unos y ceros, pero también el primer paso del algoritmo definido en [15], que calcula otra matriz de afinidad distinta. Con la eliminación de este paso y el uso de regularización para evitar el problema de las divisiones entre 0, se da por finalizado la implementación del algoritmo *Local and Global Consistency*.

La documentación lleva gran parte del tiempo de este sprint, pero se consigue ponerse al día con la memoria. Por ello la web sufre poco cambio y lo que cambia es el código con integración de métodos necesarios para los grafos.

Sprint 15

Sprint correspondiente a los días entre el 16 y el 23 de mayo. En la reunión se definen las tareas:

- Realizar estudio del funcionamiento de los algoritmos *GBILI* y *LGC* y documentarlo.

- Lectura e implementación del algoritmo RGCLI.
- Documentación del anexo B: Especificación de Requisitos.

Para finalizar con la parte de los algoritmos *GBILI* y *LGC* se decide realizar un estudio de su utilización conjunta, con varios valores en sus parámetros. Para ello, con ayuda de un mapa de calor, se realizan varias ejecuciones con distintos valores en sus parámetros, consiguiendo así localizar que parámetros afectan más y qué valores en los parámetros producen mejores resultados.

Se decide que el último algoritmo a implementar sea el *RGCLI* [2]. Es una mejora del *GBILI* por lo que la lectura y la comprensión suponen menos problema que el anterior. Se implementa una primera versión muy parecida al pseudocódigo original.

Con respecto a la documentación se extienden los requisitos ya existentes en la documentación del trabajo anterior. No hay muchos requisitos ni funcionalidades nuevas pero se considera que los definidos son muy generales y por eso se decide hacer una lista más detallada. Además, se deciden poner los casos de uso generales que entran dentro de los objetivos, entre ellos: dar opción al usuario de acceder a documentación del algoritmo, visualizar tabla de datos al cargar archivos y también visualizar una tabla de resultados finales, con la clasificación de cada dato de entrada.

Sprint 16

Sprint correspondiente a los días entre el 23 y el 30 de mayo. En la reunión se definen las tareas:

- Integración en la web de los algoritmos.
- Traducir textos generados hasta el momento.

En este tiempo el trabajo no sufre gran cambio. Mientras se intenta integrar finalmente la visualización de los grafos en la web final, se realiza un proyecto en paralelo únicamente dedicado a la visualización de grafos en una web. Para ello se intenta simular una ejecución real en este subproyecto, generando un *json* con los resultados de la ejecución, como se hacía con el *Co-Forest* y después visualizando, con ayuda de la librería *d3.js*, los grafos en cada paso. Esto lleva bastante tiempo debido a la inexperiencia con el uso de simulaciones en *javascript* y a la complejidad de representar diferentes pasos dentro de un grafo.

Sprint 17

Sprint correspondiente a los días entre el 30 de mayo y el 6 de junio. Se decide llevar el trabajo a julio y se definen las siguientes tareas:

- Corregir *RGCLI*.
- Continuar la integración en la web de los algoritmos.
- Cambiar página de inicio de la web.
- Corregir y continuar parte de anexos y memoria.

En este periodo se intenta trasladar todo el proyecto realizado aparte a la web real. Para ello se debe realizar gran parte de código específico para grafos, ya que muchos de los métodos anteriores no sirven. Buscando reutilizar la mayor parte del código, hay situaciones en las que la lógica debe ser separada entre métodos inductivos y métodos basados en grafos, pensando también en facilitar nuevas implementaciones futuras. Se consigue poder ejecutar tanto los algoritmos *GBILI* como el *RGCLI* junto con el *LGC* y se visualizan los grafos paso por paso.

Con respecto al *RGCLI*, se cambia su implementación para que no haga su ejecución con hilos y facilite la recogida de datos para representarlos.

La página de inicio de la web sufre un cambio visual. Los contenedores o *cards* que contienen los iconos de los algoritmos y que permiten seleccionarlos, se decide hacerlos de manera horizontal, además de incluir una explicación introductoria al algoritmo y permitir acceder a la documentación de cada algoritmo (en las mismas *cards*). Esto se decide así ya que de la manera en la que estaba hecho, un usuario que entra por primera vez, puede no conocer nada del algoritmo, provocando que tenga que seleccionar un algoritmo y un fichero para poder leer su explicación en la parte de configuración. Además se generan y eligen los iconos definitivos para los algoritmos basados en grafos y para el *Co-Forest*.

Sprint 18

Sprint correspondiente a los días entre el 6 de junio y 20 de junio. Se definen las siguientes tareas:

- Finalizar parte de configuración y visualización de grafos.

- Pantalla de subida de archivos.
- Crear pseudocódigos específicos para la web.
- Corregir y documentar parte de la memoria.
- Finalizar con los anexos A, B y C.

El inicio de este Sprint se utiliza para ponerse al día con la documentación, tanto de la memoria como los anexos. Para ello, se decide realizar ya todas las secciones y corregir algunas de ellas. Por ejemplo, la parte de los conceptos teóricos basados en grafos, se considera que es una parte muy importante del trabajo, y como la documentación estaba basada en un artículo el cual hace de introducción hacia estos métodos, se decide usar otro artículo como complemento, una *review* hecha en 2021 que contiene detalles más extensos y renovados [11].

En la visualización, se pretende corregir: poner las imágenes definitivas de los pseudocódigos de cada algoritmo nuevo, conseguir que en la configuración la parte de teoría cambie dinámicamente según quiera el usuario ver la fase de construcción de grafos o la fase de inferencia y por último completar el paso a paso de los algoritmos para que de forma visual se vea qué parte está ejecutando.

Además, se realiza el planteamiento que surgió en los primeros sprints de generar un resumen de los datos cada vez que se sube un archivo. Se añade también la funcionalidad de seleccionar un fichero por defecto, en lugar de tener que descargarlo.

Al inicio de este periodo se descubre *Lighthouse*, una herramienta la cual permite evaluar el rendimiento de una web. Proporciona tanto métricas como consejos de implementación para mejorar el rendimiento. Se pretende usar en lo que queda de tiempo para cumplir con los consejos y tener buenas métricas tanto en dispositivo móvil como en escritorio.

Sprint 19

Sprint correspondiente a los días entre el 20 de junio y 4 de julio. Se definen las siguientes tareas:

- Depurar y finalizar la visualización de grafos.
- Revisar todas las traducciones.

- Desplegar la web en un servidor.
- Testear web con *Selenium IDE*.
- Comprobar la integración completa con la gestión de usuarios.
- Finalizar la memoria.
- Revisar anexos del sprint anterior y finalizar los que faltan.

Este tiempo se utiliza principalmente para depurar la web y corregir errores. Primero se realiza el despliegue y después se van corrigiendo los errores que se van encontrando (mayormente traducciones y gestión de usuarios).

Una vez se considera que la web es definitiva y que no se van a realizar más cambios, se decide hacer pruebas con *Selenium IDE* a la vez que se documentan los caso de prueba.

Primero se revisa y finaliza la memoria con los nuevos cambios y por último se realizan los anexos.

Sprint 20

Sprint correspondiente a los días entre el 4 de julio y la fecha de entrega. Se utiliza para finalizar toda la documentación, tanto de memoria y anexos como del código. Se realizan los últimos cambios y se comprueba que todo está correcto.

A.3. Estudio de viabilidad

El estudio de viabilidad de un proyecto de software es una evaluación integral que analiza la posibilidad de llevar a cabo un proyecto considerando factores económicos, técnicos y legales. Este estudio ayuda a determinar si el proyecto es rentable, cumple con las regulaciones legales y es técnicamente factible. En este contexto, se abordarán específicamente la viabilidad económica, que analiza costes y beneficios, y la viabilidad legal, que examina el cumplimiento de normativas y leyes aplicables.

Viabilidad económica

La viabilidad económica en el desarrollo de software evalúa si los costes de desarrollo, mantenimiento e implementación son superados por los beneficios proyectados, abarcando tanto los costes como los beneficios potenciales.

Costes

Los costes en el desarrollo de software son los recursos económicos necesarios para llevar a cabo el proyecto, incluyendo todos los gastos relacionados con el personal, las herramientas y equipos necesarios, y otros gastos indirectos.

Costes de empleados Los costes de empleados incluyen salarios y otros gastos asociados con la contratación y retención de personal cualificado, como desarrolladores, diseñadores, gestores de proyectos y personal de soporte. En este caso solo hay un desarrollador que tiene todas las funciones anteriores y un tutor que actúa como soporte y gestor del proyecto. Teniendo en cuenta que el salario mínimo de un programador *junior* en España es de 23666€ actualizado a día 10 de junio de 2024 [5], que un trabajador con jornada completa de forma anual trabaja alrededor de 1820 horas (consultado en [3]) y que el tiempo empleado en el trabajo ha sido una media de 16 horas a la semana (contando días laborales y fin de semanas) durante 9 meses. El salario del desarrollador sale a:

$$\frac{23666\frac{€}{\text{año}}}{1820\frac{\text{horas}}{\text{año}}} \times (9\text{meses} \times 4\frac{\text{semanas}}{\text{mes}} \times 16\frac{\text{horas}}{\text{semana}}) = 7489{,}89€ \approx 832{,}21\frac{€}{mes}$$

En el caso del tutor académico se estima un tiempo medio de 1.5 hora a la semana. Si de nuevo buscamos en [6], suponemos un sueldo base promedio de 34.401€. Con las mismas suposiciones que antes, se calcula:

$$\frac{34{,}401\frac{€}{\text{año}}}{1820\frac{\text{horas}}{\text{año}}} \times (9\text{meses} \times 4\frac{\text{semanas}}{\text{mes}} \times 1{,}5\frac{\text{horas}}{\text{semana}}) = 1020{,}69€ \approx 113{,}4\frac{€}{mes}$$

Contando con que solo dos personas conforman el equipo, en la tabla A.1 se pueden ver los pagos definitivos después de aplicar los impuestos que ha de pagar la empresa [10]. Esto es un porcentaje de contingencias, de desempleo, de FOGASA y de formación profesional, respectivamente con el orden visto en la fórmula:

$$\frac{\text{Sueldo}}{1 - (0{,}236 + 0{,}055 + 0{,}002 + 0{,}006)}$$

Empleado	**Salario bruto (€/mes)**	**Coste de la empresa (€/mes)**
Desarrollador: Mario Sanz Pérez	832.21	1187.17
Supervisor	113.4	161.77
Total	945.61	1384.94
Total 9 meses	8510.49	12140.46

Tabla A.1: Salarios brutos y costes después de impuestos

Costes de *hardware* Para realizar este proyecto el equipo necesitado ha sido el reflejado en la tabla A.2. Las especificaciones del ordenador portátil, con una vida útil[1] de 4 años, son: marca *Asus VivoBook* con procesador *Intel Core i7-1065G7*, 8GB de memoria RAM y 512 GB de SSD. Como ayuda para poder agilizar el trabajo también se utiliza un monitor *Acer KA240H*. Y por úlitmo, para el despliegue de la aplicación, se utiliza el servidor más pequeño de Microsot Azure con vida útil de 1 año y un precio de 5.0881€ al mes [8].

Activo	**Coste (€)**	**Coste amortizado (€)**
Ordenador portátil	829	207.25
Monitor ayuda	105.99	26.5
Servidor	61.06	45.8
Total	1057.11	279.55

Tabla A.2: Costes para el *hardware* del proyecto

Costes de *software* : para poder desarrollar el proyecto han sido necesarios varios programas cuya adquisición y licencias son gratuitos. Algunas de las excepciones son: *Microsoft 365*, *Github Copilot* y el dominio de la página web. El coste y su amortización se ven reflejados en la tabla A.3, donde se ha considerado una vida útil de 1 año por cada activo (correspondiente al tiempo de pago).

[1]La vida útil de un dispositivo es el período durante el cual se espera que funcione de manera eficiente antes de que sea necesario reemplazarlo debido al desgaste o a la obsolescencia tecnológica.

Activo	Coste (€)	Coste amortizado (€)
Microsoft 365	69	51.75
Github Copilot	228	171
Dominio web	10.98	8.24
Total	348.75	230.99

Tabla A.3: Costes para el *software* del proyecto

Costes adicionales Además de los costes mencionados anteriormente, existen otros costes adicionales que deben considerarse. Estos incluyen la impresión y encuadernación de la documentación, así como otros costes generales de vida.

Para la impresión y encuadernación de la documentación del proyecto, se estima un coste de 50€. Los costes generales de vida, que abarcan el uso de servicios básicos como electricidad e internet durante el desarrollo del proyecto, se estiman en 100€.

En la tabla A.4 se presenta un resumen de los costes totales del proyecto, incluyendo los costes de empleados, hardware, software y otros costes adicionales.

Concepto	Coste Total (€)
Costes de empleados	12140.46
Costes de *hardware*	279.55
Costes de *software*	230.99
Costes adicionales	150
Coste Total	12701.00

Tabla A.4: Resumen de los costes totales del proyecto

Beneficios

Los beneficios de este proyecto no se miden en términos monetarios directos, dado que el objetivo principal de la web desarrollada es educativo. A continuación, se detallan los principales beneficios cualitativos y cuantitativos del proyecto.

- Accesibilidad global a recursos educativos.

- Mejora de la educación y el aprendizaje de los usuarios.
- Desarrollo de habilidades técnicas y de gestión para el desarrollador.
- Impacto positivo en la comunidad educativa.
- Potencial para futuras expansiones y colaboraciones.

Viabilidad legal

La viabilidad legal de un proyecto de software se refiere a la evaluación y aseguramiento de que el proyecto cumple con todas las leyes, regulaciones y licencias pertinentes. Esto incluye la revisión de las licencias de software utilizadas, la protección de la propiedad intelectual, y el cumplimiento de las normativas de privacidad y seguridad de datos. Es fundamental para evitar conflictos legales y asegurar que el software pueda ser utilizado, distribuido y modificado de manera legal.

En este proyecto, se ha optado por utilizar la licencia MIT [14], una de las licencias de software libre más sencillas y permisivas. Sus principales características son:

- Permite el uso comercial del software.
- Permite la modificación del software.
- Permite la distribución del software, incluyendo versiones modificadas.
- Permite sublicenciar el software.
- Permite el uso privado del software.

Bibliotecas Utilizadas

En este proyecto se utilizan diversas bibliotecas, cada una con su propia versión y licencia. En la tabla A.5 se muestran las bibliotecas utilizadas, sus versiones y sus licencias, junto con una breve descripción de cada una.

Librería	Versión	Licencia	Descripción
D3	7.8.4	ISC	Biblioteca para manipular documentos y crear gráficos.
Bootstrap	5.2.3	MIT	Framework para diseño *responsive*.
DataTables	1.10.21	MIT	*Plugin* de jQuery para tablas HTML interactivas.
jQuery	3.5.1	MIT	Biblioteca de JavaScript que simplifica el manejo del DOM.
Babel	2.14.0	BSD	Herramientas para la internacionalización.
Flask	3.0.2	BSD	Framework para aplicaciones web.
flask-babel	4.0.0	BSD	Extensión para Flask que agrega soporte para i18n.
Flask-Login	0.6.3	MIT	Manejo de sesiones de usuario para Flask.
Flask-SQLAlchemy	3.1.1	BSD	Extensión que añade soporte para SQLAlchemy a Flask.
Flask-WTF	1.2.1	BSD	Soporte para formularios en Flask.
gunicorn	21.2.0	MIT	Servidor HTTP WSGI para aplicaciones Python.
imbalanced-learn	0.12.0	MIT	Herramientas para el aprendizaje en conjuntos de datos desbalanceados.
matplotlib	3.8.2	PSF	Biblioteca para la creación de gráficos en Python.
mypy	1.9.0	MIT	Verificador de tipos para Python.
mypy-extensions	1.0.0	MIT	Extensiones para mypy.
numpy	1.26.3	BSD	Paquete para la computación científica con Python.
pandas	2.2.0	BSD	Biblioteca para el análisis y manipulación de datos.
pylint	3.1.0	GPL	Herramienta para la detección de errores y malas prácticas en Python.
scikit-learn	1.4.0	BSD	Herramientas para el análisis de datos y minería de datos.
scipy	1.12.0	BSD	Biblioteca para matemáticas, ciencias y ingeniería.
SQLAlchemy	2.0.28	MIT	Kit de herramientas SQL para Python.
Werkzeug	3.0.1	BSD	Biblioteca WSGI para desarrollo web.
WTForms	3.1.2	BSD	Biblioteca de formularios para aplicaciones web en Python.

Tabla A.5: Librerías utilizadas, versiones y licencias

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se detallan los requisitos funcionales y no funcionales del sistema. Se incluyen los casos de uso, el catálogo de requisitos y la especificación de requisitos.

B.2. Objetivos generales

Los objetivos generales del sistema se podrían resumir en:

1. Investigar sobre el aprendizaje semisupervisado: en que consisten los *ensembles* y como funcionan los algoritmos basados en grafos.

2. Implementar los algoritmos: *Co-Forest, GBILI, RGCLI, LGC*

3. Mejorar y ampliar la web de visualización de estos algoritmos que otro alumno había desarrollado el año anterior.

B.3. Catálogo de requisitos

Cuando se aborda el desarrollo de un proyecto de software, es crucial establecer una comprensión clara y organizada de los requisitos del sistema. Estos requisitos se clasifican típicamente en dos categorías principales: requisitos funcionales y no funcionales.

Los requisitos funcionales describen las funciones específicas y el comportamiento del sistema. Estos requisitos incluyen tareas, servicios o funciones que el sistema debe realizar.

Por otro lado, los requisitos no funcionales se refieren a los aspectos del sistema que definen las cualidades o atributos del sistema, como la seguridad, la usabilidad, la confiabilidad, el rendimiento y la escalabilidad. Estos no están directamente relacionados con las actividades específicas que realiza el sistema, sino con cómo se lleva a cabo esas actividades.

En este caso, al tratarse de un trabajo heredado, los requisitos son la mayoría iguales, sobretodo aquellos que tratan acerca de la gestión de usuarios, que no era el objetivo de este trabajo. Por ello, se listarán los requisitos que sí que hayan cambiado y aquellos que sean nuevos.

Requisitos funcionales

- **RF-1 Selección de algoritmo**: la aplicación debe permitir seleccionar uno de los algoritmos implementados.
    - **RF-1.1 Selección desde menú de navegación**: la opción del algoritmo se debe de poder pulsar desde la barra de navegación situada en la parte de arriba de la página web. Aunque el usuario se encuentre en cualquier otra página puede accedera ellos.
    - **RF-1.2 Selección desde página de inicio**: la otra posibilidad de acceso a los algoritmos es a través de las tarjetas situadas en la página inicial.
- **RF-2 Selección de conjunto de datos**: la aplicación debe permitir elegir un fichero `ARFF` o `CSV` para ejecutar el algoritmo.
    - **RF-2.1 Selección ya cargada**: la aplicación debe dar la opción de usar un fichero sin necesidad de descargarlo o subirlo.
    - **RF-2.2 Carga de ficheros**: debe de haber una opción que permita cargar un archivo con las extensiones definidas. Tanto arrastrando el archivo como buscándolo en el explorador de ficheros.
- **RF-3 Visualizar datos de entrada en forma de tabla**: la aplicación debe permitir visualizar los datos de entrada en forma de tabla para que el usuario tenga más contexto del conjunto de datos que está usando.

- **RF-4 Configuración del algoritmo**: la aplicación debe permitir configurar la ejecución del algoritmo con los parámetros específicos del mismo.

  - **RF-4.1 Configuración del clasificador**: el sistema debe permitir elegir que clasificador se quiere usar (si hay más de uno) y dentro de este, los parámetros que lo inician.
  - **RF-4.2 Configuración de los datos de entrada**: se debe permitir cambair los parámetros que modifican los conjuntos de datos que se usan en los algoritmos (porcentaje de etiquetados/no etiquetados, uso de reducción de dimensión, etc.).

- **RF-5 Control visualizaciones**: la aplicación debe mostrar visualizaciones interactivas: una principal (gráfico los puntos del conjunto de datos) y otra que aporta información adicional de cada paso.

  - **RF-5.1 Visualización de algoritmos inductivos**: el gráfico principal son dos ejes donde se distribuyen los puntos (estáticos) y se debe poder visualizar los cambios que sufren en cada iteración (con ayuda de un *tooltip*). La otra parte corresponde con la visualización de gráficas que muestran estadísticas de evaluación por cada iteración.
  - **RF-5.2 Visualización de algoritmos transductivos**: el gráfico principal corresponde con un grafo interactivo en el cual los nodos deben ser dinámicos. La otra parte debe estar relacionada, señalando los pasos que sigue en el pseudocódigo (construcción del grafo, inferencia, etc.) cada vez que en la visualización se pasa hacia adelante o hacia atrás.
    - **RF-5.2.1 Control de grafo**: los grafos serán interactivos, pudiendo mover cada nodo una vez se tenga el resultado final. También se permitirá hacer más llamativo un nodo para seguir su pista.
    - **RF-5.2.2 Selección entre fases o resultados**: una vez se tengan el grafo final junto con la inferencia, el usuario puede acceder tanto a los resultados viendo una matriz de confusión o a la visualización de las fases.

- **RF-6 Visualización de tutoriales**: desde cualquier página se debe poder acceder a un tutorial interactivo el cuál muestre los pasos que se deben de seguir en esa página concreta.

- **RF-7 Acceso a ayudas**: la aplicación debe dar varias opciones que ayuden al usuario a comprender el contenido de la web y a utilizarla de manera correcta.
  - **RF-6.1 Acceso a teoría y pseudocódigos**: la aplicación debe permitir visualizar el pseudocódigo junto con algo de teoría para comprender mejor el contenido.
  - **RF-6.2 Acceso al manual de usuario**.
- **RF-8 Acceder al artículo del algoritmo**: desde la página de inicio se puede acceder al artículo científico en el que se define el comportamiento de cada algoritmo concreto.

Requisitos no funcionales

Extraídos de la memoria de David [1], se han resumido y reordenado.

- **RNF-1 Disponibilidad**: el sistema de funcionar con muy alta probabilidad ante una petición y recuperarse rápido en caso de fallo.
- **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 Interoperabilidad**: el sistema debe poder utilizarse en el mayor número posible de sistemas (sistemas operativos, 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 (*Secure Sockets Layer*, SSL).
- **RNF-6 Privacidad**: el sistema debe respetar la información privada y, de ninguna forma, compartirla con terceros. Debe ser un espacio reservado confidencial con el usuario.
- **RNF-7 Escalabilidad**: el sistema debe ser capaz de crecer para ajustarse a la carga de trabajo.

- **RNF-8 Usabilidad**: el sistema debe ser altamente capaz de manejar los errores durante la ejecución (entradas erróneas, *bugs*, etc.), mostrando información precisa al usuario y recuperándos de ellos a una situación estable. Las interfaces deben ser intuitivas para facilitar al usuario sus tareas.
- **RNF-9 Internacionalización**: el sistema debe estar preparado para soportar el inglés y el español.

B.4. Especificación de requisitos

Un diagrama de casos de uso ayuda a visualizar el sistema desde la perspectiva de los usuarios finales, mostrando las diversas formas en las que interactuarán con el software. Los elementos principales de estos diagramas incluyen: actores: Representan roles de usuarios o sistemas externos que interactúan con el sistema; casos de uso: Cada caso de uso representa una secuencia específica de acciones que el sistema realiza en respuesta a una interacción con uno o más actores; relaciones: Incluyen asociaciones (líneas que conectan actores con casos de uso), inclusiones, extensiones y generalizaciones, las cuales definen cómo se relacionan y dependen entre sí los diferentes casos de uso y actores.

De nuevo, la mayoría de los casos de uso se heredan del trabajo anterior y se modifican o se crean unos pocos. Para facilitar la comprensión, en la figura B.1 se han pintado de verde aquellos que se han modificado, de azul los nuevos y de negro los que ya existían.

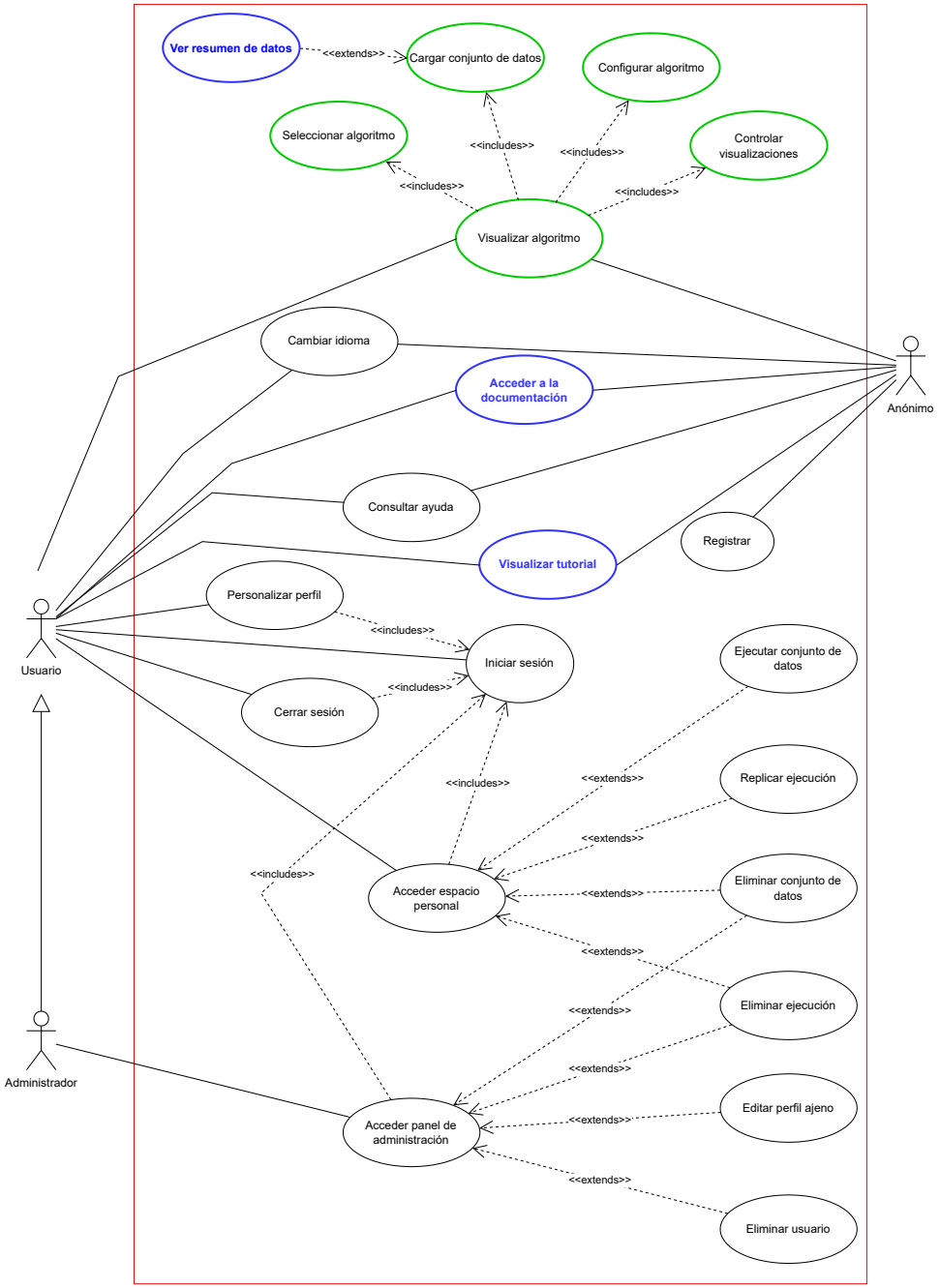

Figura B.1: Diagrama de casos de uso. En verde los modificados, en azul los nuevos y en negro los que ya existían

A continuación se definirán los casos de uso implementados y modificados en este proyecto. Para ver el resto de casos de uso, acceder a la documentación del archivo «anexos.pdf» en https://github.com/dmacha27/TFG-SemiSupervisado/tree/main/doc. Se utilizará el identificador del anterior trabajo para aquellos que se hayan modificado y se seguirá la secuencia para los nuevos.

CU-1	**Visualizar un algoritmo**
Versión	2.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-1, RF-2, RF-4, RF.5
Descripción	Visualización del proceso de entrenamiento de un algoritmo semi-supervisado. Con gráfico principal, estadísticas y fases seguidas.
Precondición	No hay precondiciones
Acciones	1. Ejecución del caso de uso 2 (Seleccionar algoritmo). 2. Ejecución del caso de uso 3 (Cargar conjunto de datos). 3. Ejecución del caso de uso 4 (Configurar algoritmo). 4. Se realiza la ejecución interna del algoritmo, obtención de la información y creación de los gráficos. 5. El usuario verá en la página los distintos gráficos de su visualización. Opcional Ejecución del caso de uso 5 (Controlar visualizaciones). Opcional Ejecución del caso de uso 19 (ver tablas con datos clasificados)
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 (modal) y al cerrarlo volverá al paso 3. ▪ La ejecución del algoritmo finalizó con excepciones (paso 4), serán capturadas y se volverá al paso 3.
Importancia	Alta

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

CU-2	**Seleccionar algoritmo**
Versión	2.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-1, RF-8
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. El área de acción del *mouse* es toda la tarjeta excepto el botón reservado para acceder a la documentación del articulo (Caso de uso 20). 3. Se redirige al usuario a la página de subida.
Postcondición	Algoritmo almacenado en su sesión y redirección a la página de subida.
Excepciones	Sin excepciones
Importancia	Alta

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

CU-3	**Seleccionar conjunto de datos**
Versión	2.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-2, RF-3
Descripción	Carga en el sistema o selección 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	1. El usuario tiene dos opciones de selección de un fichero: cargar uno del equipo o seleccionar uno ya cargado. 2. Se establece en la sesión el fichero seleccionado. 3. Se muestra una tabla interactiva con los datos del fichero.
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: Caso de uso 3: Seleccionar conjunto de datos.

CU-4	**Configurar algoritmo**
Versión	2.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-4
Descripción	Parametrización de la ejecución del algoritmo. Cada algoritmo tiene sus propios parámetros.
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.
Extensiones	a En el caso de los grafos, el usuario podrá elegir entre la teoría de la fase de construcción del grafo o la fase de inferencia. b El algoritmo que esté seleccionado en la tarjeta de configuración es el que se mostrará en la teoría.
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 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 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. ▪ El formulario está incompleto o erróneo. En este caso se bloqueará el envío del formulario indicando qué se debe arreglar.
Importancia	Alta

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

CU-5	**Controlar visualizaciones (grafos)**
Versión	1.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-5.2
Descripción	Manipulación de las visualizaciones de forma interactiva.
Precondición	Ejecutando el caso de uso 1, haber seleccionado la tarjeta de grafos en el caso de uso 2 y haber ejecutado ya el caso de uso 4.
Acciones	1. El usuario verá la visualización principal así como las fases seguidas. 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 nodos del grafo, moviéndolos, clicándolos o realizando su inferencia. 6. El usuario puede ver las estadísticas de las predicciones.
Postcondición	El usuario ha podido manejar con libertad los elementos mostrados.
Excepciones	Sin excepciones
Importancia	Alta

Tabla B.5: Caso de uso 5: Controlar visualizaciones (grafos).

CU-19	**Acceder a documentación del algoritmo**
Versión	1.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-1.2, RF-8
Descripción	Acceso a página web que contiene el artículo científico que define el algoritmo.
Precondición	Sin precondiciones.
Acciones	1. El usuario puede acceder al artículo científico del algoritmo pulsando el botón de color gris de la tarjeta de inicio. 2. Se abrirá una pestaña aparte en el navegador.
Postcondición	El usuario ha accedido a la web que contiene el artículo relacionado.
Excepciones	Sin excepciones
Importancia	Baja

Tabla B.6: Caso de uso 19: Acceder a documentación del algoritmo

CU-20	**Ver tabla con resumen de datos de entrada**
Versión	1.0
Autor	Mario Sanz Pérez
Requisitos asociados	RF-2, RF-3
Descripción	Visualización en forma de tabla de los datos de entrada.
Precondición	Ejecutando el caso de uso 1 y haber ejecutado ya el caso de uso 2 y 3 (hay fichero en la sesión).
Acciones	1. Se mostrará una tabla con los datos de entrada. 2. El usuario podrá buscar entre los datos, ordenarlos o filtrarlos. 3. El usuario podrá mostrar las entradas que quiere ver en la tabla.
Postcondición	El usuario puede interactuar con la tabla de datos.
Excepciones	▪ Si el fichero no es un ARFF o CSV, se mostrará un mensaje de advertencia en lugar de la tabla. ▪ Si el fichero no se puede leer por el formato, se mostrará un mensaje de error en lugar de la tabla.
Importancia	Media

Tabla B.7: Caso de uso 20: Ver tabla con resumen de datos de entrada.

CU-21	**Visualizar tutoriales**
Versión	1.0
Autores	David Martínez Acha y Mario Sanz Pérez
Requisitos asociados	RF-6
Descripción	Visualización de tutoriales en cada ventana.
Precondición	Sin precondiciones.
Acciones	1. En la página de inicio: el tutorial consta de 9 pasos. 2. En la página de selección de datos: el tutorial consta de 7 pasos. 3. En la página de configuración: en caso de inductivos consta de 5 pasos y en caso de grafos de 6. 4. En la página de visualización: si son métodos inductivos consta de 6 pasos tanto si son métodos de grafos como inductivos. 5. En cualquier paso del tutorial, este se puede cerrar. 6. Ver el siguiente o el anterior paso.
Postcondición	El usuario puede seguir navegando por la página en la que estaba
Excepciones	Sin excepciones
Importancia	Media

Tabla B.8: Caso de uso 21: Visualizar tutoriales.

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este anexo proporciona una documentación detallada sobre la programación del proyecto. Aquí se abordan aspectos cruciales como la estructura de directorios, los pasos para la compilación, instalación y ejecución del proyecto, así como las pruebas del sistema. Este documento está dirigido a desarrolladores y contribuyentes que deseen entender y trabajar con el código fuente del proyecto.

D.2. Estructura de directorios

La estructura de directorios del proyecto es fundamental para entender cómo está organizado el código y dónde se encuentran los diferentes componentes. A continuación se muestra una representación visual de la estructura de directorios:

TFG-Semi-Supervised-Learning/:
ruta padre del proyecto

algoritmos/:
algoritmos de aprendizaje semisupervisado implementados. Cada uno de ellos está programado como una clase, para luego usarlo como un objeto en la aplicación web.

experimentos:
contiene notebooks de Jupyter con experimentos y pruebas de los algoritmos de grafos. Puede ser ignorado ya que tiene un propósito de depuración.

utilidades:
utilidades que realizan ciertos pasos de la aplicación y de los algoritmos (comunes).

docs/:
documentación del proyecto hecha con LaTeX.

img:
imágenes utilizadas en la documentación (tanto en la memoria como en la documentación técnica)

tex:
archivos LaTeX de la memoria y anexos separados por secciones.

web/:
aplicación web de la versión 2.0 del proyecto.

app/:
contiene el código fuente de la aplicación web.

___init___.py:
inicializa la aplicación y configura las rutas através del método `create_app()`.

datasets/:
directorio donde se almacenan los archivos que se usarán en la web.

anonimos/:
archivos anónimos subidos por los usuarios.

registrados/:
archivos subidos por los usuarios registrados.

seleccionar/:
archivos para probar la aplicación.

static/:
contiene los archivos estáticos de la aplicación (CSS, JS, imágenes).

css/:
contiene los estilos CSS de la aplicación. En un único archivo `style.css`.

js/:
contiene los scripts de JavaScript de la aplicación.

configuración/:
scripts de ventana configuración de los algoritmos.

usuarios/:
scripts de ventanas de usuarios.

visualización/:
scripts de ventanas de visualización.

json/:
contiene el archivo de parámetros de configuración de los algoritmos.

pseudocodigos/:
imágenes de los pseudocódigos de los algoritmos (en formato PNG para la web y PDF para el usuario).

runs/:
directorio donde se almacenan los resultados de las ejecuciones de los algoritmos (en formato JSON).

templates/:
contiene las plantillas HTML de la aplicación.

configuracion/:
plantillas de configuración de los algoritmos.

usuarios/:
plantillas de las ventanas de usuarios.

visualizacion/:
plantillas de las ventanas de visualización.

translations/: es/: LC_MESSAGES/:
contiene los archivos de traducción de la aplicación, el .po será el archivo de traducción y el .mo el archivo compilado.

instance:
contiene la base de datos de la aplicación (SQLite).

tests:
contiene el proyecto de pruebas de la aplicación web, hecho con Selenium IDE.

ficheros:
ficheros que se utilizan en las pruebas, uno de error y otro válido.

venv/:
entorno virtual de Python.

D.3. Manual del programador

En esta sección se comentarán aquellos aspectos que se consideran más relevantes para que un programador pueda entender el *backend* de la aplicación. Para ello, se explicará el proceso típico de adición de un algoritmo, dejando de lado la gestión de usuarios en la que se centra más el trabajo [1].

La primera fase consiste en la implementación del algoritmo. Hay que tener en cuenta que la versión de Python utilizada es la 3.11 (la más actual hasta el inicio del proyecto). Se debe añadir a la carpeta `algoritmos` el archivo que contendrá la clase que representa al algoritmo. En el caso de que sea inductivo, los métodos más típicos se deben denominar: `fit` y `predict`, para que cuando se lanze desde flask, todo esté automatizado. El método fit, a la vez que va ejecutando el algoritmo, debe ir almacenando en una estructura (preferiblemente *DataFrame*) los datos que se van a representar en la visualización. Para ello el programador debe saber en que lugar y momento del código debe almacenar los datos.

En el otro caso, con los métodos transductivos, existen dos posibles algoritmos, el de fase de construcción del grafo y el de fase de inferencia de etiquetas. De nuevo cada uno separado en un archivo que contendrá la definición de una clase. Los algoritmos de construcción de grafos vistos en artículos científicos generalmente reflejan una solución en forma de conjunto de enlaces o en forma de matriz de pesos. Esto no nos interesa en este caso, por lo que se debe buscar la manera de organizar el algoritmo en varias fases (las que se representarán en la web) y devolver los grafos de cada momento o fase. Es decir, lo que devolverá el método de `construirGrafo` serán varios diccionarios en representación a cada fase del algoritmo. Estos diccionarios deben tener la forma Número de nodo: [Conjunto de vecinos enlazados], ... En cuanto a la fase de inferencia, el algoritmo que se quiera implementar, siempre va a necesitar el grafo final construido, para localizar sus enlaces y tenerlos en cuenta. El algoritmo de inferencia depende de cada caso, pero se debe tener un método final que prediga todas las etiquetas de los datos que no estaban etiquetados. En el caso del *Local and Global Consistency* no tiene sentido separar por fases los resultados, ya que puede

tener miles de iteraciones variando un mismo resultado esas mil veces, lo que no aporta información valiosa al usuario. En caso de que el algoritmo se pueda dividir en fases, habría que pensar en cómo almacenar los datos para su representación.

Una vez se tienen los algoritmos implementados, es hora de utilizarlos en la web. Para ello, se deben crear los elementos estáticos, como la tarjeta de selección inicial, la opción en el menú de navegación, el pseudocódigo tanto en inglés como en español y un icono representativo. La parte previa a la ejecución de estos nuevos algoritmos será la de configuración. Para ello es importante destacar que existe un archivo llamado `parametros.json` el cual contiene todos los posibles parámetros de configuración de los clasificadores (en caso de los inductivos) y de los algoritmos (en caso de los transductivos) (ver figura D.1).

Figura D.1: Estructura archivo `parametros.json`

Se debe distinguir en que clave de entre *Inductive*, *Inference* o *Graphs* se deben incluir los parámetros. Otro detalle importante es que para traducir estos parámetros, como otro recurso diferente a *Babel*, se utiliza un diccionario propio.

Una vez se tienen estos parámetros, se debe seguir el proceso de crear o modificar los métodos:

- En `configuration_routes`: en el método `configurar_algoritmo` se debe crear el formulario correspondiente a cada uno.
- En `visualization_routes`: se debe crear un método `parametros_algoritmoX`: recogerá los parámetros introducidos en el formulario de configuración y se los pasa a la plantilla HTML.
- En la plantilla HTML del algoritmo, se debe crear una llamada al método `inicializar` o a métodos que hagan lo propio. Este método realiza una petición POST a una ruta no visualizada donde ejecuta los algoritmos. Los parámetros anteriores se pasarán a este nuevo plano y los tendrá en cuenta.
- En `data_routes`: crear método `datosAlgoritmoX` el cuál llamará, según su tipo, al método que devuelve toda la información (en formato JSON) de la ejecución. Estos métodos que obtiene los datos instancian la clase del algoritmo y obtienen su resultado. También se encargan de procesar los resultados. Es decir, en los grafos, es necesario sacar la información de: nodos, enlaces, predicciones y mapa de clases; y en los inductivos se deberá obtener: iteraciones, log, estadísticas y mapa de clases.

Una vez se tienen los datos procesados, se pueden usar en el javascript y HTML para crear los gráficos y elementos que se crean oportunos.

D.4. Compilación, Instalación y Ejecución del Proyecto

En este apartado se detallan los pasos necesarios para compilar, instalar y ejecutar el proyecto VASS2.

Instalación

Para instalar el proyecto, se deben seguir los siguientes pasos en una terminal del equipo:

1. Clonar el repositorio:

```
git clone https://github.com/msp1015/TFG-Semi-Supervised-Learning
```

2. Navegar al directorio del proyecto:

```
cd TFG-Semi-Supervised-Learning
```

3. Crear un entorno virtual:

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

4. Activar el entorno virtual:

   - En Windows:

     ```
     .\venv\Scripts\activate
     ```

   - En macOS/Linux:

     ```
     source venv/bin/activate
     ```

5. Instalar dependencias:

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

6. Creación de directorios necesarios:

```
cd web/app
mkdir runs
```

   - En Windows:

     ```
     mkdir datasets\anonimos
     mkdir datasets\registrados
     ```

   - En macOS/Linux:

     ```
     mkdir datasets/anonimos
     mkdir datasets/registrados
     ```

7. Compilar traducciones:

```
pybabel compile -d translations
```

Uso

Para ejecutar la aplicación, siga los siguientes pasos:

1. Iniciar la aplicación:

```
cd ..
flask run
```

2. Abrir su navegador y navegar a `http://localhost:5000` para acceder a la aplicación.

3. (Opcional) Añadir `-debug` al final del comando `flask run` para entrar en modo desarrollo.

Despliegue con Gunicorn y Nginx en servidor

Para desplegar la aplicación en un entorno de producción usando Gunicorn y Nginx en un servidor, siga los siguientes pasos:

1. Instalar Gunicorn y Nginx en su máquina de servidor:

```
sudo apt update
sudo apt install python3-gunicorn nginx
```

2. Navegar al directorio del proyecto (creado con los pasos vistos en el punto anterior) e iniciar Gunicorn en el puerto 8000:

```
cd /ruta/a/TFG-Semi-Supervised-Learning/web
gunicorn -b 0.0.0.0:8000 'app:create_app()'
```

3. Configurar Nginx para servir la aplicación. Abra el archivo de configuración de Nginx:

```
sudo nano /etc/nginx/sites-available/default
```

4. Añadir la siguiente configuración en el archivo de Nginx:

```
server {
    listen 80;
    server_name tu_dominio.com;

    location / {
        proxy_pass http://0.0.0.0:8000;
    }
}
```

5. Guardar y cerrar el archivo, luego reiniciar Nginx:

```
sudo systemctl restart nginx
```

6. Asegurarse de que Gunicorn y Nginx están ejecutándose correctamente:

```
sudo systemctl status nginx

sudo systemctl status gunicorn
```

Internacionalización con Babel

Para añadir soporte de internacionalización con Babel, siga los siguientes pasos:

1. Extraer los mensajes a traducir:

```
pybabel extract -F babel.cfg -o messages.pot
```

2. (Opcional) Si utiliza el marcador `lazy_gettext` en lugar de `gettext`, extraer los mensajes de la siguiente manera:

```
pybabel extract -F babel.cfg -k lazy_gettext -o messages.pot .
```

3. Actualizar el catálogo de traducciones:

   ```
   pybabel update -i messages.pot -d translations
   ```

   A continuación de este paso se debe traducir el archivo `messages.po` en la carpeta `translations`. Además de traducir a mano los textos nuevos, es importante fijarse en aquellos con la etiqueta «fuzzy», ya que pueden generar problemas de traducción y no serán incluidos en la aplicación. Una vez comprobado que estas traducciones estan correctas, se elimina la etiqueta «fuzzy».

4. Compilar las traducciones:

   ```
   pybabel compile -d translations
   ```

D.5. Pruebas del sistema

Esta sección del proyecto se centra en las pruebas de la aplicación web para garantizar la calidad y fiabilidad del sistema. Se ha optado por utilizar Selenium IDE debido a sus múltiples ventajas, siendo una heramienta poderosa y fácil de usar e instalar.

En el marco de este proyecto, se ha decidido heredar todos los casos de prueba definidos en el trabajo anterior, garantizando así la continuidad y la cobertura de pruebas ya establecidas. Y se han añadido nuevos casos de prueba para cubrir las nuevas funcionalidades implementadas en la versión 2.0. Se ha necesitado redefinir los anteriores para adaptarlos al nuevo dominio y a una ruta válida de subida de archivos.

En la figura D.2 se muestra la lista de pruebas que se han hecho en Selenium IDE, organizados en *Test Suites*.

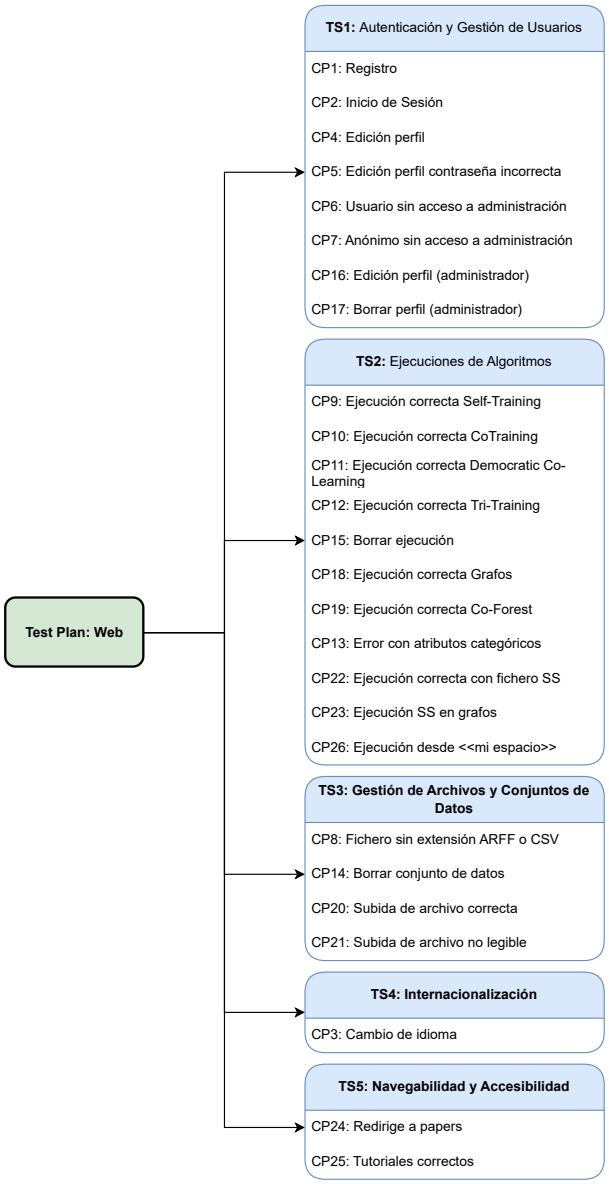

Figura D.2: Conjunto de test organizados en Test Suites

ID	CP-18
Prioridad	Alta
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Ejecución completa de Grafos
Precondiciones	Sistema en funcionamiento
Datos Utilizados	*Dataset* proporcionado
Pasos	1. Acceder a vass2.dev/. 2. Seleccionar la opción de Grafos. 3. Seleccionar los datos necesarios para la ejecución. 4. Seleccionar la opción «Configurar Algoritmo». 5. Comprobar redirección a página de configuración de grafos. 6. Seleccionar el botón «Ejecutar». 7. Comprobar redirección a página de resultados comparando el título. 8. Comprobar existencia de leyenda. 9. Ejecutar los n pasos hasta la inferencia.
Resultados esperados	Ejecución exitosa del algoritmo de Grafos, resultados mostrados correctamente.
Estado	Exitoso

Tabla D.1: CP18: Ejecución correcta y completa de grafos.

ID	CP-19
Prioridad	Alta
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Ejecución correcta de Co-Forest
Precondiciones	Sistema en funcionamiento
Datos Utilizados	*Dataset* para Co-Forest
Pasos	1. Acceder a vass2.dev/. 2. Seleccionar la opción de Co-Forest. 3. Seleccionar los datos necesarios para la ejecución. 4. Seleccionar la opción «Configurar Algoritmo». 5. Comprobar redirección a página de configuración del Co-Forest. 6. Seleccionar el botón «Ejecutar». 7. Comprobar redirección a página de resultados comparando el título. 8. Comprobar existencia de leyenda. 9. Comprobar la existencia de estadísticas específicas.
Resultados esperados	Ejecución exitosa del algoritmo Co-Forest, resultados mostrados correctamente
Estado	Exitoso

Tabla D.2: CP18: Ejecución correcta y completa de Co-Forest.

ID	CP-20
Prioridad	Alta
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Subida de archivo correcta con usuario registrado
Precondiciones	Sistema en funcionamiento, archivo válido disponible y usuario en sesión
Datos Utilizados	Archivo válido en formato ARFF `iris.arff` y usuario de prueba
Pasos	1. Iniciar sesión en el sistema. 2. Seleccionar cualquier algoritmo. 3. Seleccionar la opción para subir un nuevo archivo del equipo. 4. Confirmar la subida del archivo. 5. Confirmar la aparición de la tabla con los datos del archivo subido. 6. Seleccionar el botón «Mi espacio». 7. Comprobar que existe una entrada en la tabla de datos del archivo subido.
Resultados esperados	Archivo subido correctamente, guardado en el sistema y mostrado en forma de tabla
Estado	Exitoso

Tabla D.3: CP20: Subida de archivo con usuario registrado

ID	CP-21
Prioridad	Media
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Subida de archivo no legible sin usuario en sesión
Precondiciones	Sistema en funcionamiento, archivo no legible disponible
Datos Utilizados	Archivo no legible `error.txt`
Pasos	1. Seleccionar cualquier algoritmo. 2. Seleccionar la opción para subir un nuevo archivo del equipo. 3. Confirmar la subida del archivo. 4. Confirmar la aparición de un mensaje de advertencia en lugar de la tabla de datos. 5. Seleccionar el botón «Configurar Algoritmo». 6. Comprobar redirección a página de inicio. 7. Confirmar aparición de mensaje de error.
Resultados esperados	El sistema muestra un mensaje de error indicando que el archivo no es legible y redirige a la página de inicio
Estado	Exitoso

Tabla D.4: CP21: Subida de archivo ilegible

ID	CP-22
Prioridad	Alta
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Ejecución correcta con fichero SS
Precondiciones	Sistema en funcionamiento, fichero SS disponible
Datos Utilizados	Fichero *Breast Cancer* semisupervisado
Pasos	1. Acceder a vass2.dev/. 2. Seleccionar la opción de Co-Forest. 3. Seleccionar archivo *Breast Cancer* de prueba. 4. Seleccionar la opción «Configurar Algoritmo». 5. Comprobar redirección a página de configuración del Co-Forest. 6. Bajar al mínimo el porcentaje de datos no etiquetados (debe ignorarlo). 7. Seleccionar el botón «Ejecutar». 8. Comprobar redirección a página de resultados comparando el título. 9. Comprobar existencia de leyenda. 10. Comprobar la existencia de estadísticas específicas.
Resultados esperados	Ejecución exitosa del algoritmo con fichero SS, resultados mostrados correctamente
Estado	Exitoso

Tabla D.5: CP22: Ejecución con fichero semisupervisado

ID	CP-23
Prioridad	Alta
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Ejecución SS en grafos
Precondiciones	Sistema en funcionamiento, datos de grafos disponibles
Datos Utilizados	Datos de grafos para SS
Pasos	1. Acceder a vass2.dev/. 2. Seleccionar la opción de Grafos. 3. Seleccionar archivo *Breast Cancer* de prueba. 4. Seleccionar la opción «Configurar Algoritmo». 5. Comprobar redirección a página de configuración del Co-Forest. 6. Bajar al mínimo el porcentaje de datos no etiquetados (debe ignorarlo). 7. Seleccionar el botón «Ejecutar». 8. Comprobar redirección a página de resultados comparando el título. 9. Comprobar existencia de leyenda. 10. Pulsar n veces hasta la inferencia. 11. Seleccionar el botón «Inferir etiquetas». 12. Comprobar texto de advertencia en resultados (no se puede inferir etiquetas con datos de entrada SS).
Resultados esperados	Ejecución exitosa del algoritmo SS con grafos, mensaje de advertencia mostrado correctamente.
Estado	Exitoso

Tabla D.6: CP23: Ejecución de con fichero semisupervisado en grafos

ID	CP-24
Prioridad	Baja
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Redirige a artículos científicos
Precondiciones	Sistema en funcionamiento
Datos Utilizados	No aplica
Pasos	1. Acceder a vass2.dev/. 2. Por cada tarjeta de selección: Seleccionar la opción «Artículo». 3. Verificar la redirección a 6 diferentes pestañas en el navegador.
Resultados esperados	El usuario es redirigido correctamente a las 6 páginas científicas.
Estado	Exitoso

Tabla D.7: CP24: Redirección a artículos científicos en la web

ID	CP-25
Prioridad	Media
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Tutoriales correctos en todas las páginas
Precondiciones	Sistema en funcionamiento, botón de «tutorial» existe
Datos Utilizados	No aplica
Pasos	1. Acceder a vass2.dev/. 2. Seleccionar la opción «Tutorial». 3. Comprobar texto de cada paso del turorial de inicio. 4. Seleccionar un algoritmo, ejemplo: Co-Forest. 5. Seleccionar la opción «Tutorial». 6. Comprobar texto de cada paso del tutorial de subida de archivo. 7. Seleccionar la opción «Configurar Algoritmo». 8. Seleccionar la opción «Tutorial». 9. Comprobar texto de cada paso del tutorial de configuración de algoritmo. 10. Seleccionar la opción «Ejecutar». 11. Seleccionar la opción «Tutorial». 12. Comprobar texto de cada paso del tutorial de resultados.
Resultados esperados	Los tutoriales se muestran correctamente y son accesibles para el usuario (tanto anónimo como registrado)
Estado	Exitoso

Tabla D.8: CP25: Ejecución de tutoriales en todas las ventanas.

ID	CP-26
Prioridad	Media
Fecha de Ejecución	03/07/2024
Tester	Mario Sanz Pérez
Descripción	Ejecución desde «mi espacio»
Precondiciones	Usuario autenticado, acceso a «mi espacio», ejecución previa almacenada en base de datos
Datos Utilizados	Usuario de Prueba, ejecución almacenada con fichero X
Pasos	1. Acceder a «mi espacio» desde la interfaz de usuario. 2. Seleccionar el algoritmo deseado en la tabla de historial. 3. Comprobar redirección a página de resultados comparando el título. 4. Comprobar existencia de leyenda. 5. Comprobar la existencia de estadísticas específicas.
Resultados esperados	Ejecución exitosa del algoritmo desde «mi espacio», resultados mostrados correctamente
Estado	Exitoso

Tabla D.9: CP26: Ejecución desde cuenta de usuario registrado.

Apéndice E

Documentación de usuario

E.1. Introducción

Esta sección proporciona una visión general de la aplicación web, describiendo su propósito y las principales funcionalidades que ofrece. Se detallan los requisitos necesarios para poder usarla.

E.2. Requisitos de usuarios

En esta sección se enumeran los requisitos mínimos que los usuarios deben cumplir para utilizar la aplicación web de manera efectiva. Esto incluye:

- Navegador web compatible: Los usuarios deben tener un navegador moderno (como Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) actualizado a la última versión.
- Conexión a Internet: Se requiere una conexión a Internet estable y de banda ancha para acceder y utilizar todas las funcionalidades de la aplicación web.
- Sistema operativo: La aplicación web es compatible con los principales sistemas operativos (Windows, macOS, Linux, Android, iOS).

E.3. Instalación

No es necesario llevar a cabo ninguna instalación para usar la aplicación web, ya que está disponible en línea. Sin embargo, para los desarrolladores

que deseen contribuir al proyecto o modificar el código, se proporciona el enlace al repositorio de GitHub. Instrucciones sobre cómo clonar el repositorio y configurar el entorno de desarrollo se encontrarán en el anexo D del documento https://github.com/msp1015/TFG-Semi-Supervised-Learning/blob/main/doc/anexos.pdf.

E.4. Manual del usuario

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

Credenciales administrador

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

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:

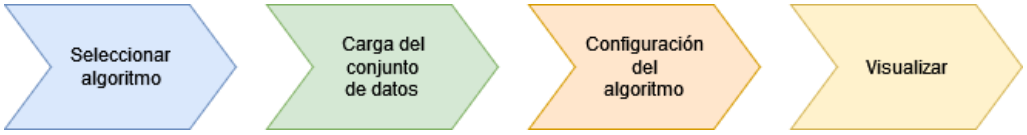

Figura E.1: Flujo de la visualización de un algoritmo

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 (ver figura E.2). Para saber qué algoritmo se está seleccionando, se marca en azul la opción del menú de navegación, esto indica que la sesión contiene un algoritmo seleccionado.

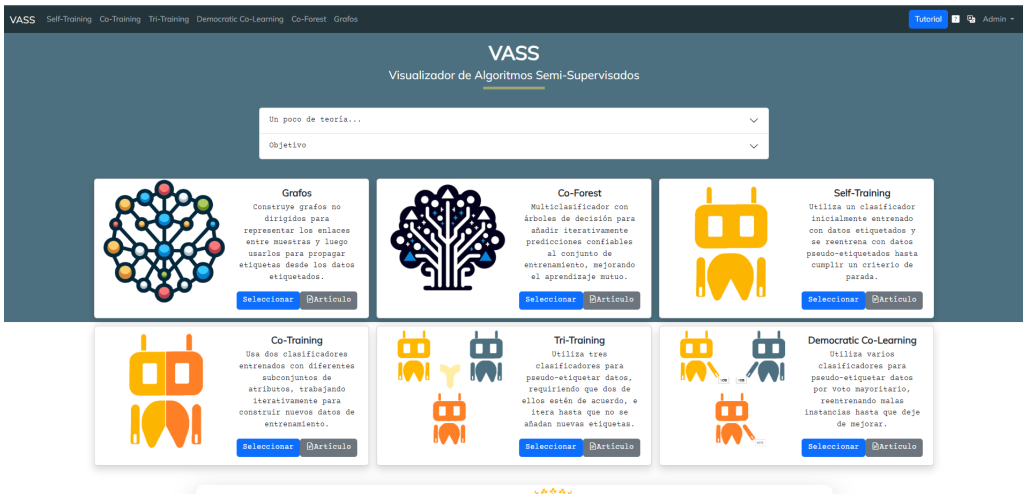

Figura E.2: Página principal

Todo el área de la tarjeta actúa como botón de selección, y una vez se haya seleccionado, será redirigido a la página de subida del fichero.

Otra característica de estas tarjetas es que tiene un botón secundario con el cual se puede acceder al artículo científico (o al medio que lo publica) que define el comportamiento de cada algoritmo. Se abrirá una ventana nueva por lo que la sesión seguirá en el navegador.

Carga del conjunto de datos

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

Figura E.3: Carga del conjunto de datos

Si el usuario no dispone de un fichero, la aplicación incluye cuatro posibilidades para probar la aplicación. Pulsando en el botón «Usa un fichero de prueba» se establecerá en la sesión el fichero `iris.arff`, y pulsando en el desplegable, se podrá elegir entre las opciones *Iris*, *Breast Cancer*, *Breast Cancer (SS)* o *Diabetes*.

La carga o selección de un fichero implica que la sesión del usuario, esté registrado o no, tenga un fichero establecido por defecto para siguientes ejecuciones.

Si hay un fichero establecido en la sesión, aparecerá su contenido en forma de tabla interactiva, como se puede ver en la figura E.4. En ella se podrá filtrar por orden cada fila y buscar un elemento en concreto.

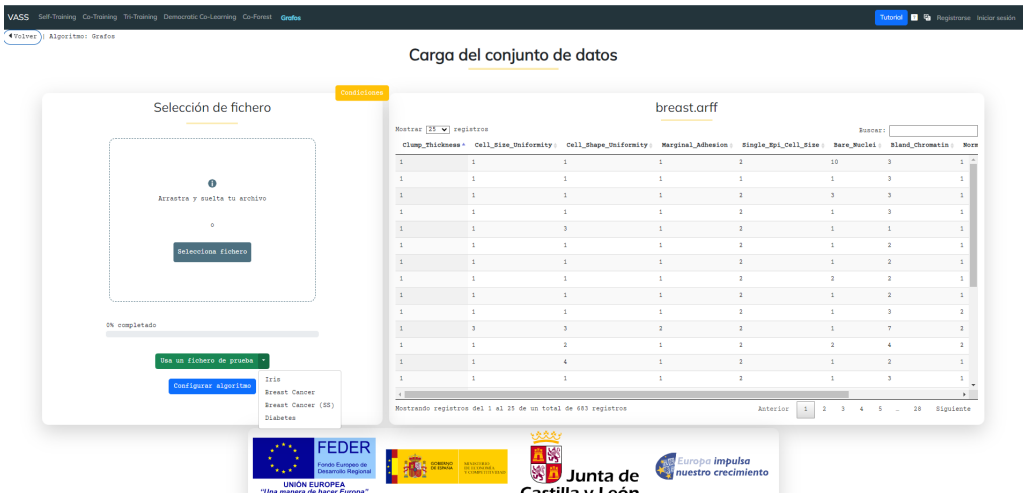

Figura E.4: Selección de ejemplo y vista de tabla

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

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

Requisito fundamental

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

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

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

Las condiciones de entrada vienen especificadas al pulsar en el botón de la esquina superior derecha del recuadro de selección de fichero «Condiciones». Al pulsar en él aparecerá una ventana emergente con dichas condiciones (ver figura E.5).

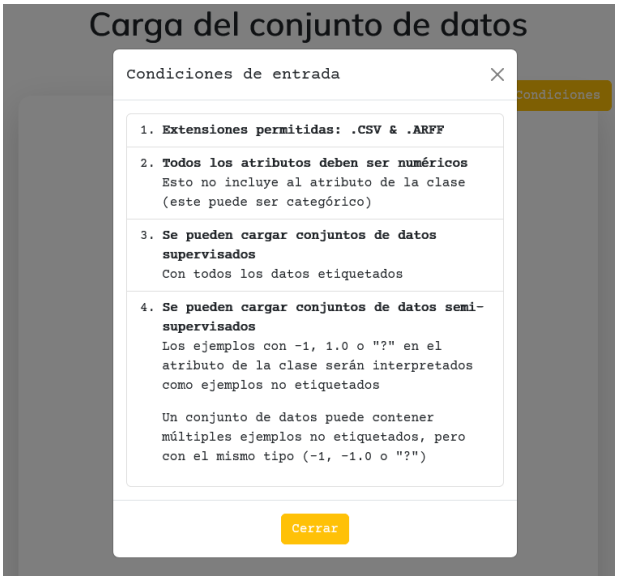

Figura E.5: Condiciones de entrada

En el caso de que se suba un fichero con la extensión correcta pero que tenga un formato no legible o cuando directamente la extensión no es correcta, se advertirá al usuario en el lugar donde iría la tabla con datos, como se observa en la figura E.6.

Figura E.6: Mensaje de advertencia en caso de carga de datos errónea

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

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

Configuración del algoritmo

Se encontrará en la página de configuración del algoritmo, donde podrá observar un apartado teórico con sus conceptos generales y su pseudocódigo. Por otro lado, tendrá un formulario con todos los parámetros que se pueden configurar para ese algoritmo (ver figura E.7).

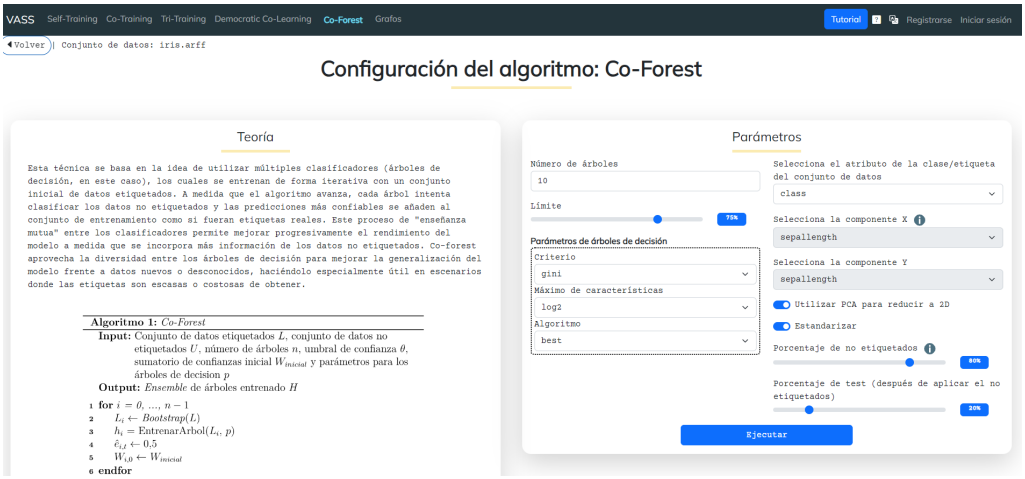

Figura E.7: Configuración del algoritmo

Todos los parámetros tienen establecido un valor por defecto (con la configuración «estable»), pero se tiene libertad completa para modificar cada uno de ellos. El atributo «clase» se establece por defecto al valor de la última columna del conjunto de datos, ya que por defecto suele estar colocada siempre en esta posición. El usuario puede ejecutar directamente sin cambiar ningún valor en el caso que lo desee.

Esta pantalla sigue un estándar para todos las posibles selecciones, pero en el caso de los grafos (donde el algoritmo implementado se selecciona en la configuración), la pantalla de la teoría cambia mínimamente (ver figura E.8). Se puede cambiar entre la teoría de inferencia y la de construcción del grafo,

y el contenido dentro de este también cambiará dinámicamente según el algoritmo que esté seleccionado en el formulario de parámetros.

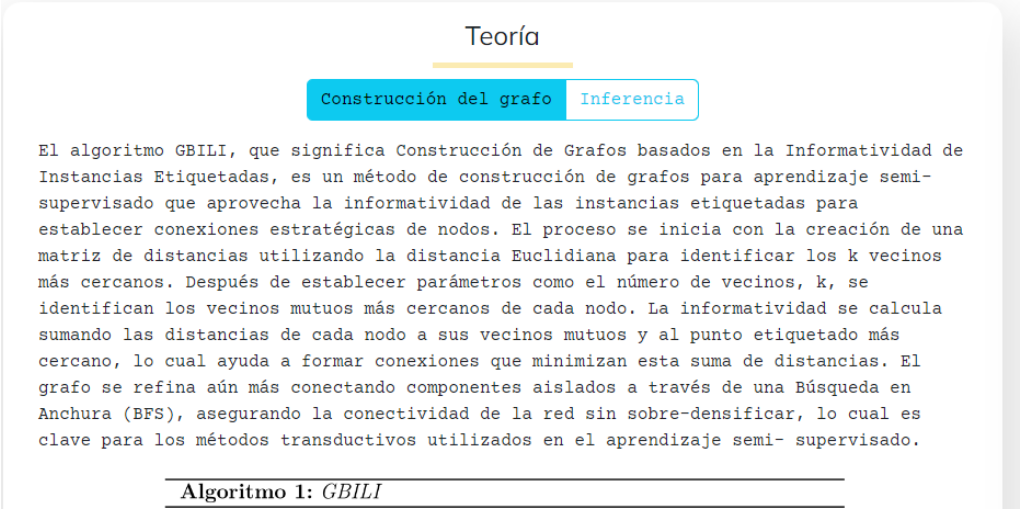

Figura E.8: Tarjeta de teoría en la configuración de grafos

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

Visualización

En la página de visualización conviene separar los métodos basados en grafos de los inductivos. Esto es porque, aunque la idea sea la misma, la visualización y su interacción es completamente diferente.

Aun así, existen elementos comunes como una animación de carga mientras se ejecuta el algoritmo y dos tarjetas con la representación de la respuesta, a la izquierda se mostratá la visualización y a la derecha los resultados.

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

Visualización principal de algoritmos inductivos En el gráfico principal se mostrará el conjunto de datos en dos dimensiones. Aquí podrá verse qué es lo que ocurre durante el proceso de entrenamiento del algoritmo (ver E.9).

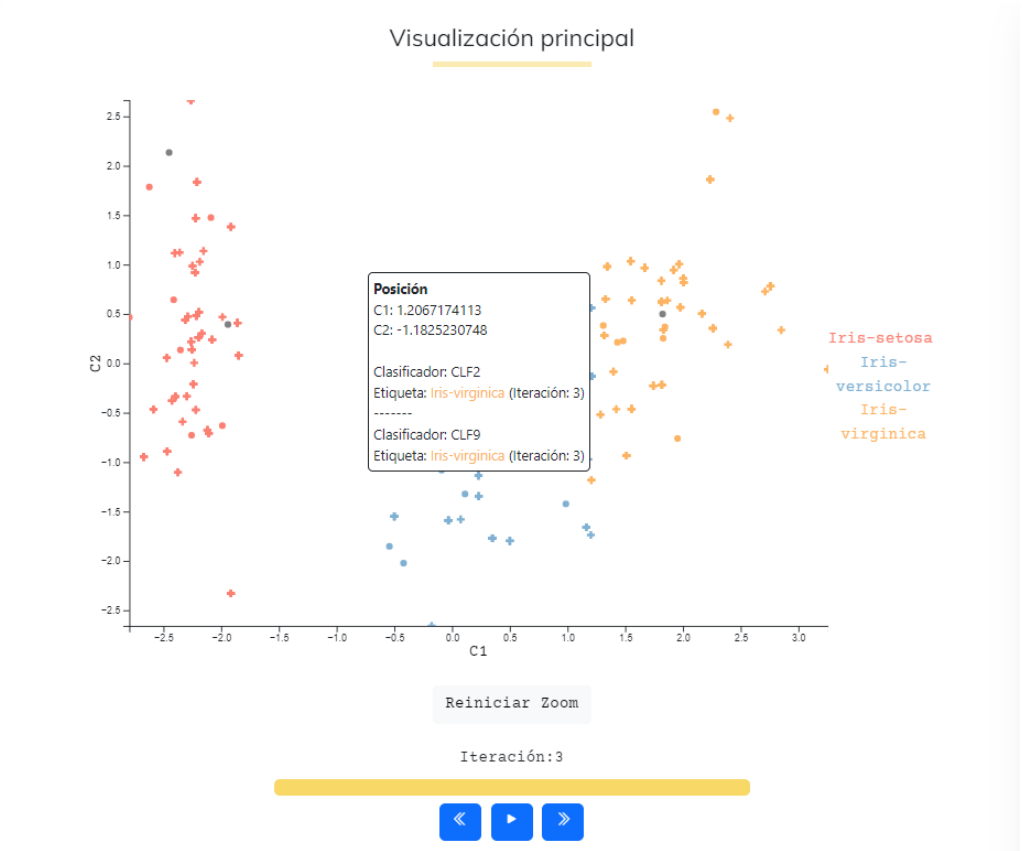

Figura E.9: Visualización principal

Este gráfico es interactivo:

- Permite realizar `zoom` sobre una zona deseada mediante el doble clic o moviendo la ruleta del ratón (ampliando si es táctil).
- 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 E.9.

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  .
- Avanzar iteración manualmente con el botón  .
- Retroceder al paso previo con el botón  .

Todas estas acciones modificarán el estado de los gráficos.

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

Tooltip: Casos comunes Existe un formato común a todos los algoritmos para los casos de los datos iniciales. Este tipo de puntos se representa mediante un círculo. Como dato inicial, tendrá la etiqueta correspondiente. Además, cada punto tiene en la parte superior la posición que ocupa en el gráfico. Esta posición coincide con los atributos seleccionados para representar los datos. En la figura E.10 se presenta un ejemplo de todo lo anterior donde se seleccionó PCA y por eso aparece «C1» y «C2».

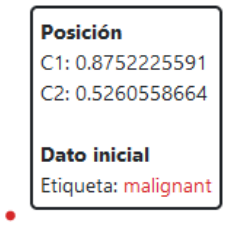

Figura E.10: *Tooltip* con dato inicial

Otro formato común es en el caso de puntos solapados (derivados de ejemplos duplicados en el conjunto de datos o por PCA). En este caso, donde en el anterior aparecía la información del punto, ahora aparecerá un listado con todos los puntos solapados (ver figura E.11).

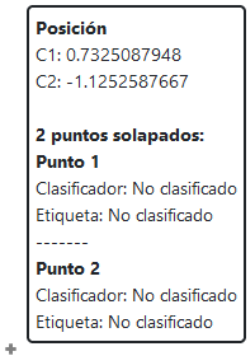

Figura E.11: *Tooltip* con datos solapados

Como se puede ver en este ejemplo de la figura E.11, los puntos no han sido clasificados (captura tomada en iteración cero), esto no es importante, la diferencia con una iteración posterior es que aparecerá la etiqueta asignada (se verá en cada algoritmo). Además, aparece un indicativo de «Clasificador», se ha querido incluir en este ejemplo porque en todos los algoritmos excepto *Self-Training* se indica el clasificador que se ha encargado de etiquetar cada punto.

Es interesante comentar que puede darse el caso de puntos solapados en el que alguno o todos sean datos iniciales (ver figura E.12).

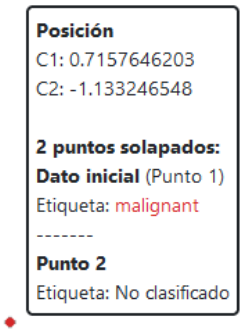

Figura E.12: *Tooltip* con datos solapados con uno inicial

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

Existen dos formatos, el primero es en el que el dato no ha sido etiquetado (porque se etiqueta en una iteración posterior o simplemente porque nunca

es etiquetado). Por ejemplo, una captura tomada en la iteración dos de una dato no etiquetado es la de la figura E.13. En dicho ejemplo puede ser extraño que no aparezca un título indicando el punto (algo como «Punto 1»), ese formato solo se «activa» cuando existen datos solapados.

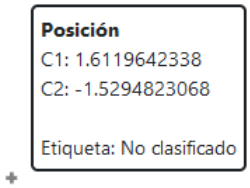

Figura E.13: *Tooltip* con un dato no clasificado

El otro formato es en el que el dato ya ha sido clasificado (en la iteración actual o en una previa). Por ejemplo, una captura tomada en la iteración ocho de una dato etiquetado en la iteración seis es la de la figura E.14.

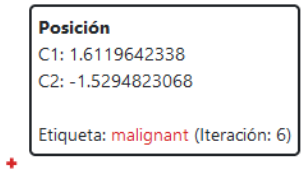

Figura E.14: *Tooltip* con un dato clasificado

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

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

Vuelven a existir dos formatos, cuando el dato no ha sido etiquetado todavía (o nunca, ver figura E.15) y cuando sí está etiquetado.

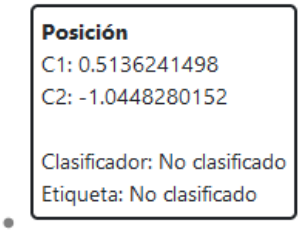

Figura E.15: *Tooltip* con un dato no clasificado

Para el caso del dato clasificado sí que es necesario puntualizar alguna cuestión (se verá en el ejemplo siguiente). *Co-Training* considera dos clasificadores base y cada uno de ellos puede clasificar a un punto. Esto se ha representado de dos maneras. La primera señal es que el símbolo del punto será uno concreto para cada clasificador. La segunda señal es que en el *tooltip* aparecerá una línea de «Clasificador:» que estará seguida de dicho símbolo y el nombre del clasificador (ver figura E.16). Se mantiene también el número de la iteración en la que se ha clasificado.

Figura E.16: *Tooltip* con un dato clasificado

Esta idea también es la misma si existen datos solapados. En ese caso, en cada uno de los puntos que se indiquen en el listado de solapados aparecerá el clasificador (el símbolo y el nombre) junto con la etiqueta (ver figura E.17).

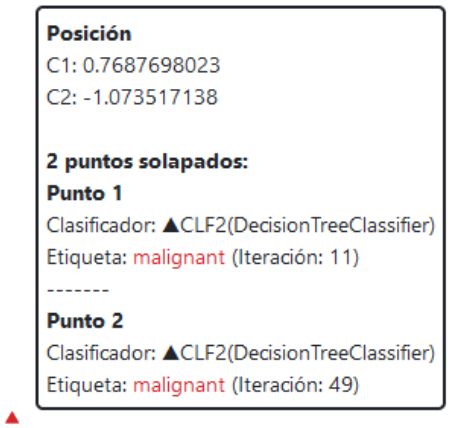

Figura E.17: *Tooltip* con datos solapados y clasificados

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

La particularidad que tiene *Democratic Co-Learning* y *Tri-Training* es que cada dato puede ser etiquetado por varios clasificadores. Para ejemplificar esto simplemente se realiza un listado de los clasificadores que lo han etiquetado (ver figura E.18).

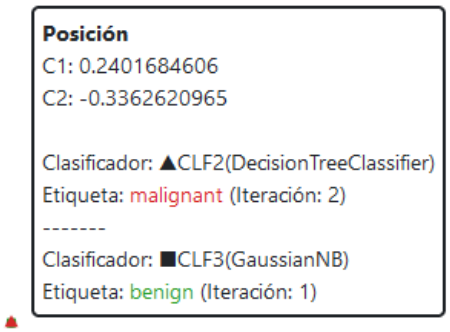

Figura E.18: *Tooltip* con un dato clasificado por dos clasificadores

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

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

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

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

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

Por último, como en todos los anteriores formatos, pueden existir datos solapados. El ejemplo mostrado en la figura E.19 es de *Democratic Co-Learning*, pero como se ha comentado, es el mismo formato que el de *Tri-Training*.

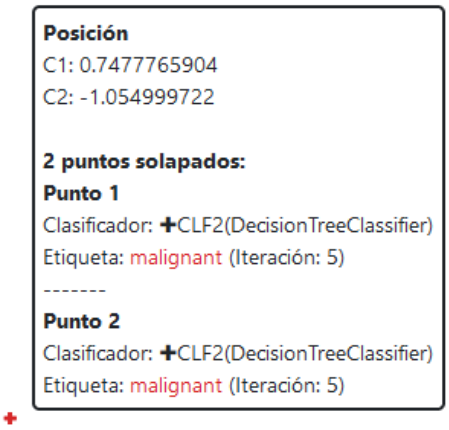

Figura E.19: *Tooltip* con datos solapados y clasificados

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

Tooltip*: *Co-Forest el caso del algoritmo *Co-Forest* es muy parecido al de *Democratic Co-Learning*. Un dato puede ser clasificado por varios árboles de decisión en cada iteración, ya que cada árbol tiene su propio conjunto de entrenamiento. No es igual que *Tri-Training* ya que un clasificador no puede

clasificar dos veces al mismo dato. En este caso desaparece el dibujo que identifica cual es cada clasificador, ya que únicamente se utilizan árboles de decisión. Por ello directamente se identifican por «CLF» y el número del árbol, como se puede ver en la figura E.20.

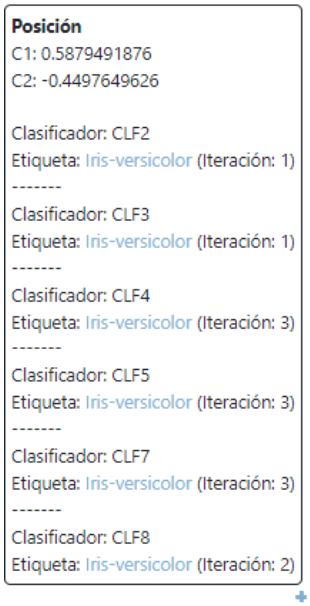

Figura E.20: *Tooltip* de ejecución final del Co-Forest

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

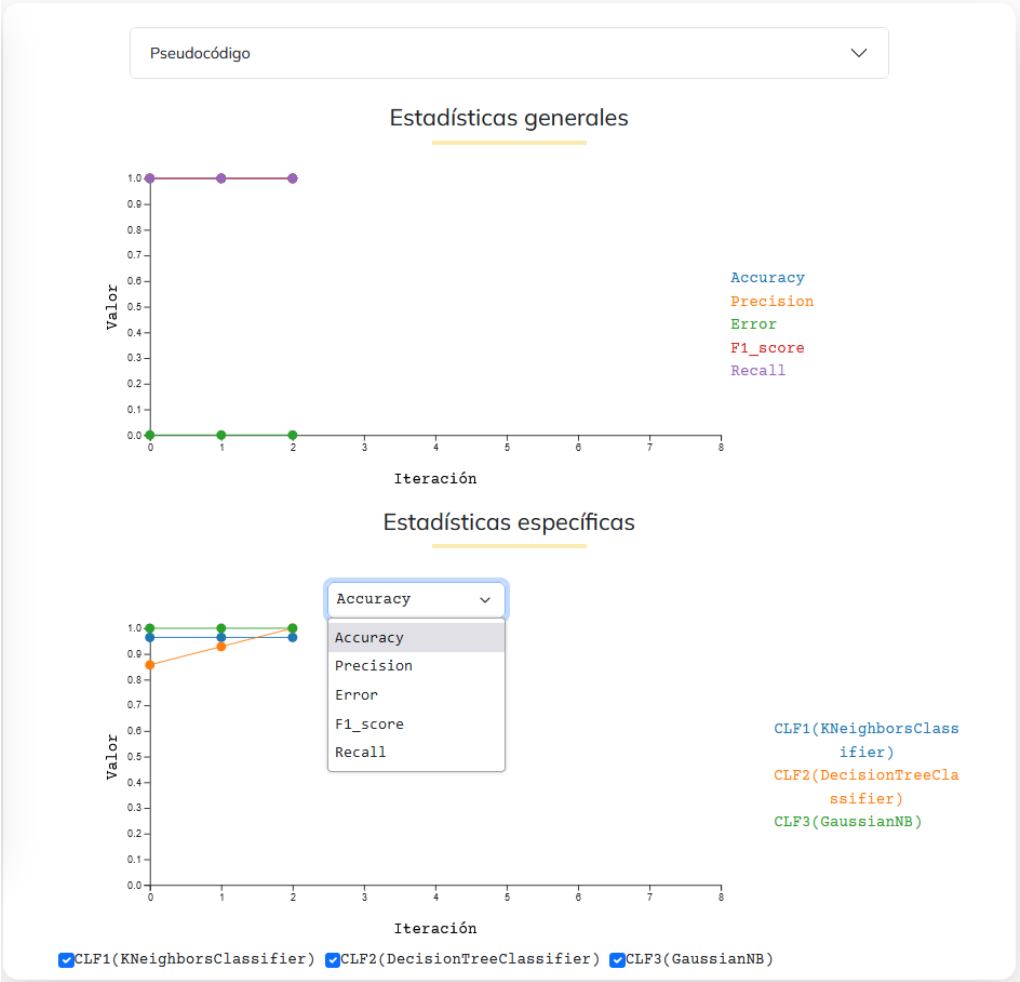

Figura E.21: Gráficos estadísticos

En la parte superior de esta zona se tiene un desplegable (contraído por defecto) que contiene el pseudocódigo (el mismo que en la fase de configuración). Si se desea consultar, simplemente se pulsa en cualquier parte del desplegable (ver figura E.22).

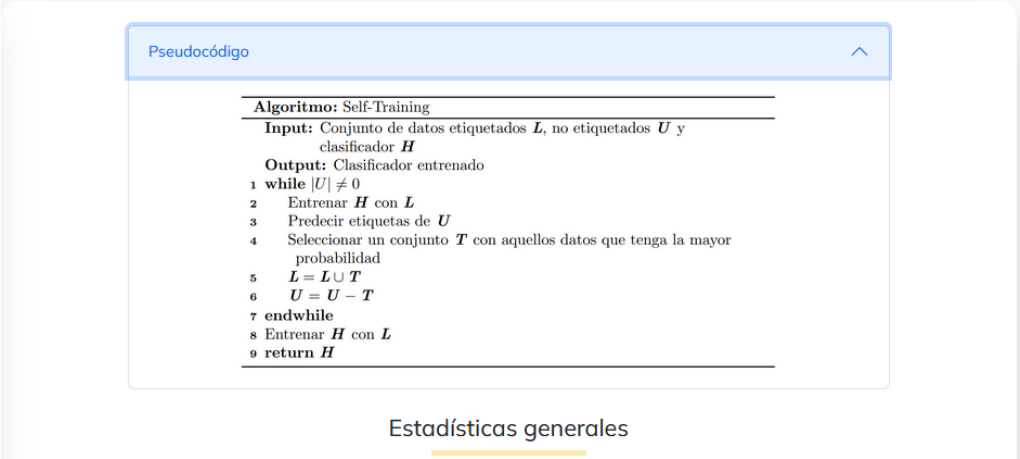

Figura E.22: Desplegable pseudocódigo

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

En el caso de las estadísticas específicas puede seleccionar qué estadística mostrar mediante el selector superior, y de qué clasificadores mostrarla mediante las casillas en la parte inferior (ver figura E.21).

Un caso particular es el gráfico de estadísticas específicas del *Co-Forest*, ya que es común utilizar muchos árboles de decisión, puede que el usuario no interprete bien la información, y para facilitar la visualización existen dos selectores que permiten marcar todos los clasificadores o no marcar ninguno (ver figura E.23)

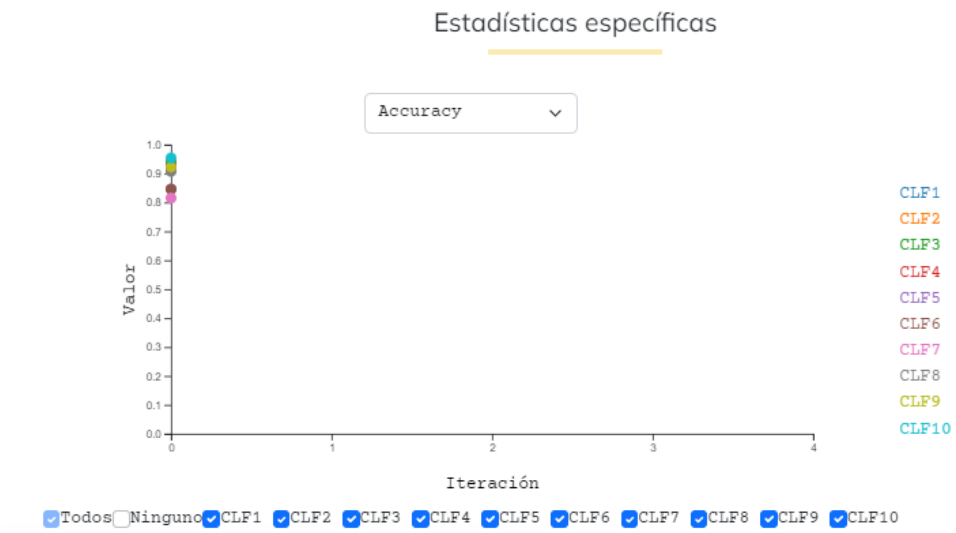

Figura E.23: Estadísticas específicas en *Co-Forest*

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

Visualización principal de algoritmos transductivos Entramos ahora en la otra posible visualización de algoritmos: los grafos. Como se ha comentado, la división de la pantalla es igual, pero con contenido diferente (ver figura E.24).

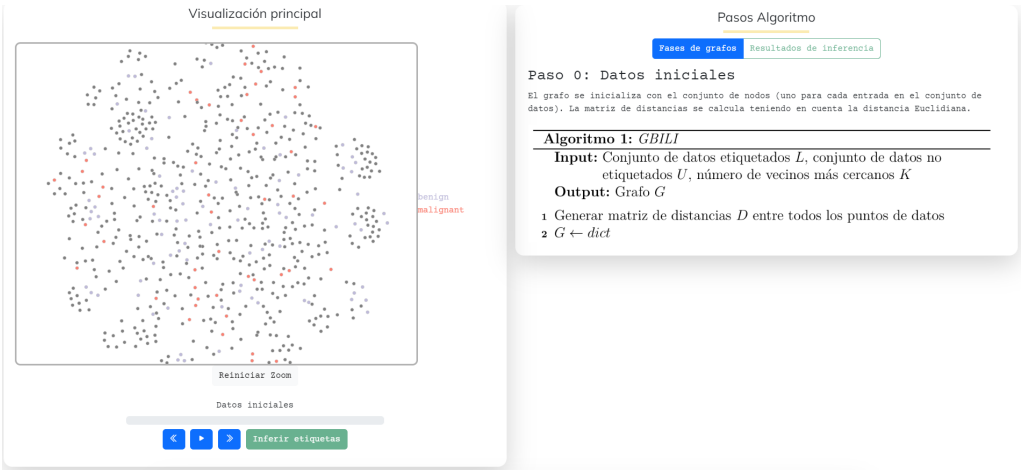

Figura E.24: Página de visualización de grafos

La visualización principal se puede manejar usando los mismos botones que para los algoritmos inductivos, pero con un botón añadido.

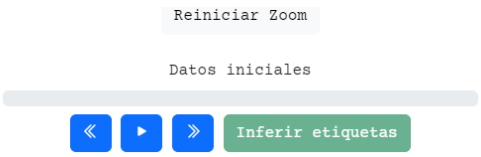

Figura E.25: Controles para grafos

Además de poder hacer zoom, el botón nuevo se puede ver en la figura **??**. Este únicamente estará disponible al llegar al último paso del algoritmo. Detallar que el texto que aparece será la fase o paso del algoritmo actual en ese momento.

La visualización principal muestra un grafo no dirigido representado por nodos y enlaces (ver figura E.26).

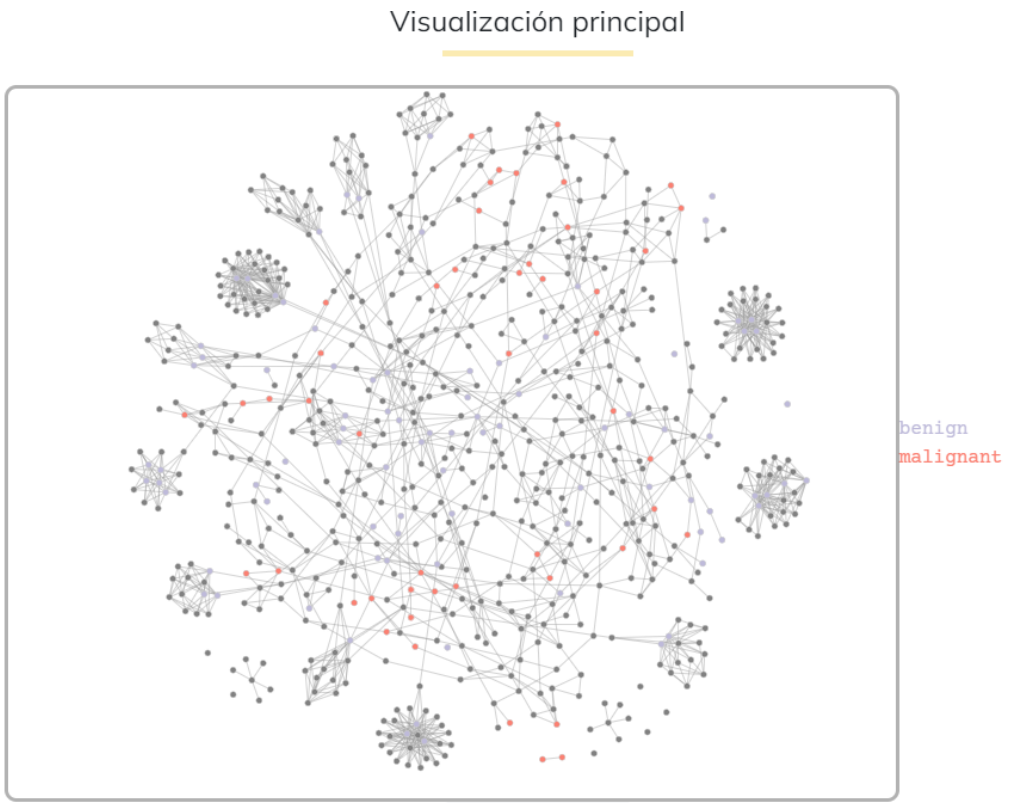

Figura E.26: Visualización principal del grafo

Los nodos del grafo pueden ser pulsados para que resalte tanto él mismo, como los enlaces y sus vecinos en cada paso. Esto permite al usuario localizar cuales son los vecinos de un nodo en grafos que están muy «apelotonados». Un clic en cualquier otra parte del gráfico quitará esta selección. Cuando se pasa de fase, si se mantiene pulsado, el nodo se actualizará con sus vecinos actuales (ver figura E.27).

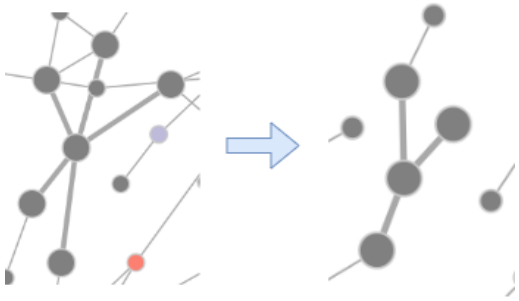

Figura E.27: Nodo seleccionado en dos fases seguidas del algoritmo

En la última iteración se permite al usuario «jugar» con el grafo, pudiendo arrastrar nodos y con ellos a sus vecinos. Además, como se ha comentado antes, se habilitará el botón de «Inferir etiquetas» visto en la figura E.25. Al pulsar este botón cambiarán los valores de las etiquetas de los nodos no clasificados con una transición (ver figura E.28), además de habilitar la sección que se explicará a continuación.

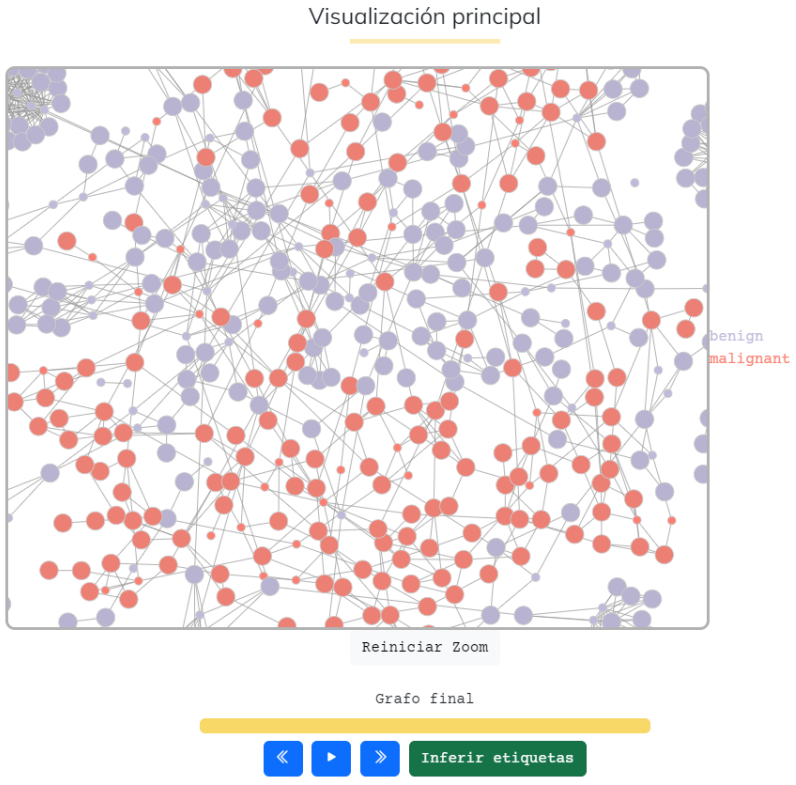

Figura E.28: Instante de inferencia sobre los nodos

Como último detalle, la leyenda en este caso es interactiva, y si se pulsa en cualquiera de las clases o etiquetas, se verán los datos iniciales con esa clase, esté en la fase que esté el algoritmo, únicamente mostratá los iniciales, como es el caso de la figura E.29, en el que se han inferido las etiquetas y posteriormente se ha pulsado en una clase.

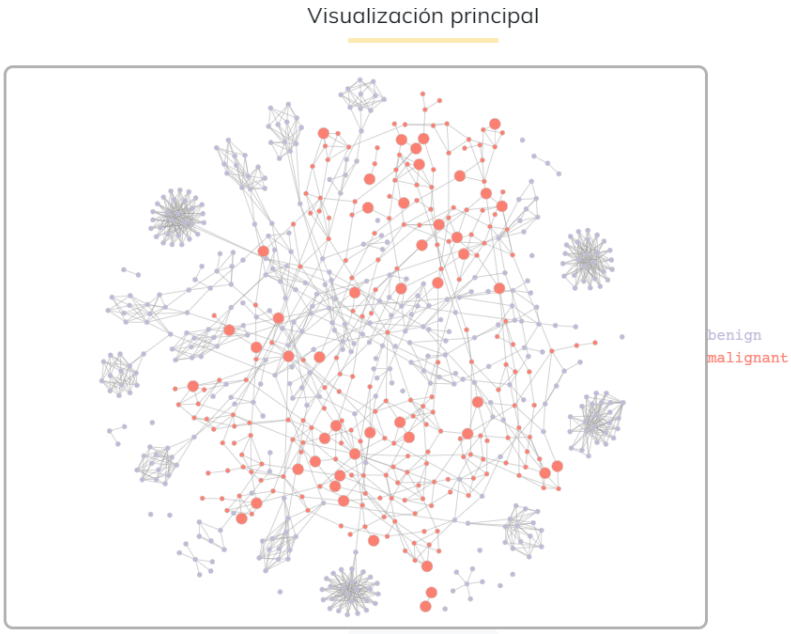

Figura E.29: Datos iniciales de una clase

La sección de la parte derecha de la pantalla, vista en la figura E.24, inicialmente contendrá la explicación de cada fase junto con su pseudocódigo. De manera que a medida que el usuario vaya avanzando, este contenido cambie. Al inferir las etiquetas se mostrará lo siguiente en la parte derecha:

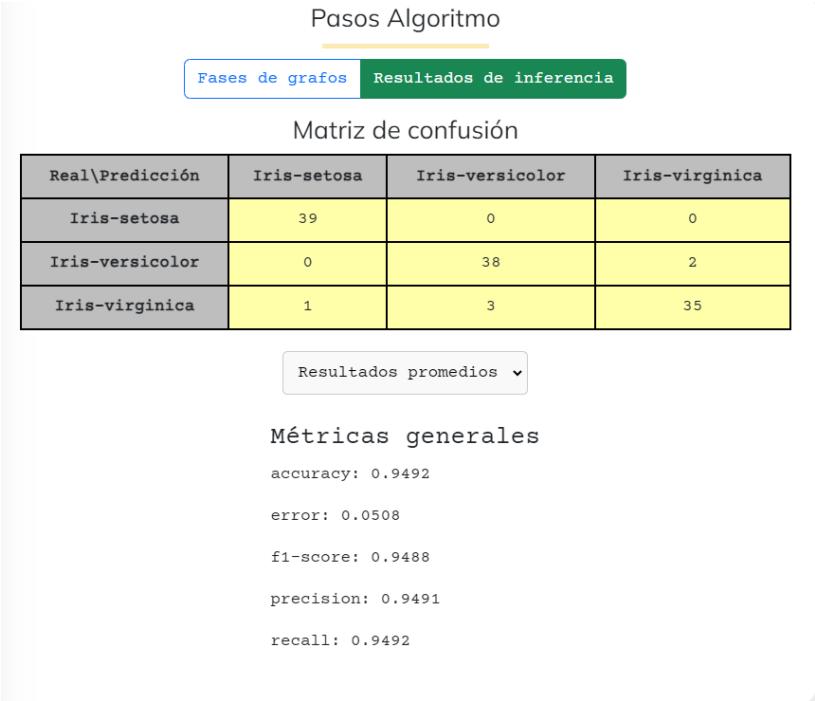

Figura E.30: Resultados de inferencia en grafos

Se considera importante destacar que esta sección se ha cambiado con respecto a los inductivos porque los grafos no contienen una fase de test, lo que hace que la evaluación tenga que ser sobre los datos que no estaban etiquetados. Inicialmente se ve la matriz de confusión y el cálculo de sus métricas, pero se puede cambiar para ver las métricas específicas de cada clase (ver figura E.31)

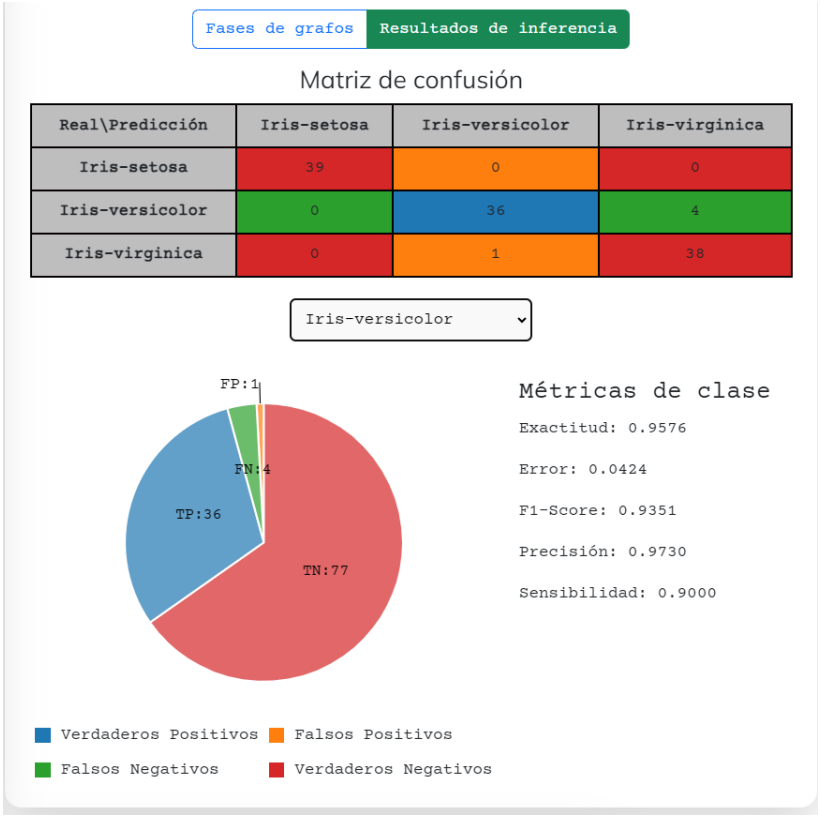

Figura E.31: Resultados de inferencia de una clase concreta

Indicar que los colores del gráfico de tarta son representativos con los de la matriz de confusión, representando así cómo se calcula cada valor del gráfico.

Tutoriales y cambio de idioma

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

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

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

Estas funciones pueden verse en la figura E.32.

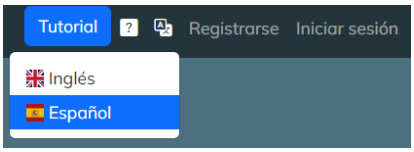

Figura E.32: Cambiar de idioma

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

Además de estas funciones, en la figura E.32 se observa un botón de «Tutorial». Al pulsar en él se abrirá una ventana emergente con varios pasos (ver figura E.33). Cada paso se puede avanzar o retroceder mediante los botones de la parte inferior, y en cualquier momento se podrá cerrar. Se ha incluido un tutorial en todas las páginas de la aplicación, resumiendo la funcionalidad de cada ventana en una serie de pasos. Se podrá acceder al tutorial siempre que se desee.

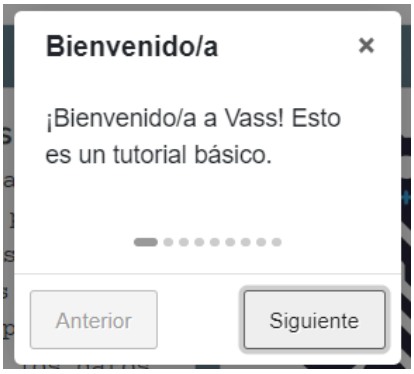

Figura E.33: Ejemplo de tutorial

Registrarse

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

Para acceder a él, en la barra de navegación se dispone de un enlace (en la parte derecha) que redirecciona al formulario de creación (ver figura E.34). 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.34: Formulario de registro

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

Iniciar sesión

Al igual que para el registro, para iniciar sesión existe un enlace en la barra de navegación (en la parte derecha) que redirecciona al formulario de inicio de sesión (ver figura E.35).

Figura E.35: Formulario de inicio de sesión

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

Cerrar sesión

Si ya tiene sesión iniciada, debe hacer clic en su nombre en la barra de navegación (parte derecha). Esto abrirá un desplegable en el que, aparte de otras opciones, podrá cerrar la sesión (ver figura E.36).

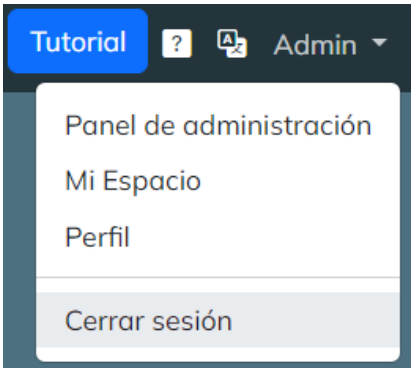

Figura E.36: Cierre de sesión

Personalizar perfil

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

En primer lugar, y similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Perfil» (ver figura E.37).

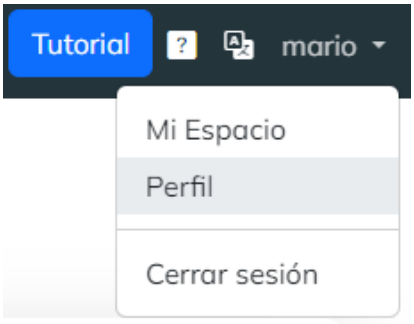

Figura E.37: Acceso al perfil

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

La parte de edición (zona derecha) contiene un formulario similar al del registro (ver figura E.38). 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.38: Perfil personal y edición

Espacio personal

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

En primer lugar, y similar a otros casos, en el desplegable de la barra de navegación del usuario se pulsa en «Mi Espacio» (ver figura E.39).

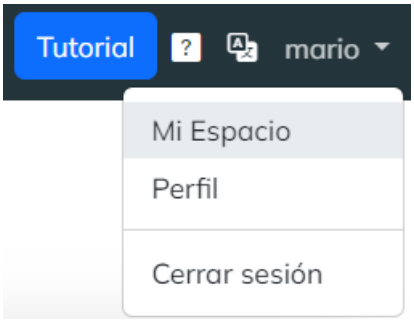

Figura E.39: Acceso al espacio personal

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

En la parte derecha se encontrarán dos tablas en las que se reflejan los ficheros subidos y las ejecuciones (ver figura E.40).

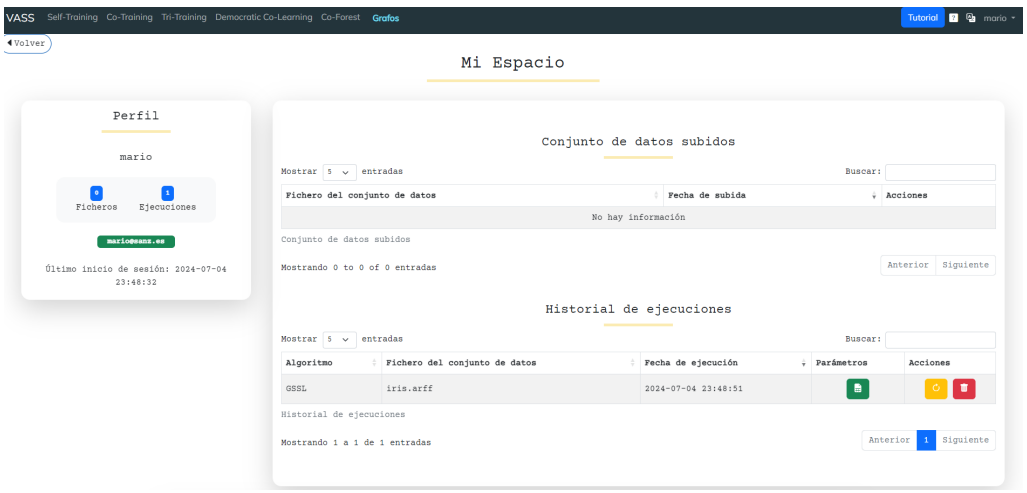

Figura E.40: Espacio personal

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

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

En el caso de querer utilizar el fichero para **ejecutar un algoritmo**, simplemente se ha de pulsar en el botón  . Al hacerlo, se mostrará una ventana emergente (`modal`) para seleccionar el algoritmo deseado (ver figura E.41).

Figura E.41: Selección de algoritmo

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

Por otro lado, si se quiere **eliminar un fichero** de la cuenta (y del sistema), se pulsa en el botón  . De igual manera, se mostrará una ventana emergente de confirmación (ver figura E.42).

Figura E.42: 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 (columna parámetros). Esto mostrará una ventana emergente con un JSON legible y formateado (ver figura E.43).

Figura E.43: 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 (símbolo típico de recargar página).

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

Exactamente igual a los conjuntos de datos, las ejecuciones pueden **eliminarse** pulsando en el botón mostrando una ventana emergente de confirmación similar a la anterior (ver figura E.44).

Figura E.44: Eliminar ejecución

E.5. Manual del administrador

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

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

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

Panel de administración

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

En primer lugar, para acceder al panel, similar a otros casos, en el desplegable de la barra de navegación del usuario (con rol administrador) se pulsa en «Panel de administración» (ver figura E.45).

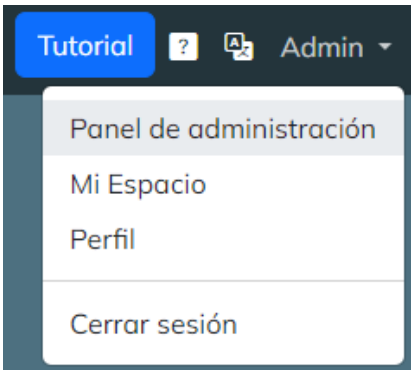

Figura E.45: Acceso al panel de administración

Una vez dentro, se tiene un menú organizado en pestañas donde ver las tres tablas (ver figuras E.46 (Administración de usuarios), E.47 (Conjuntos de datos subidos) y E.48 (Historial de ejecuciones)).

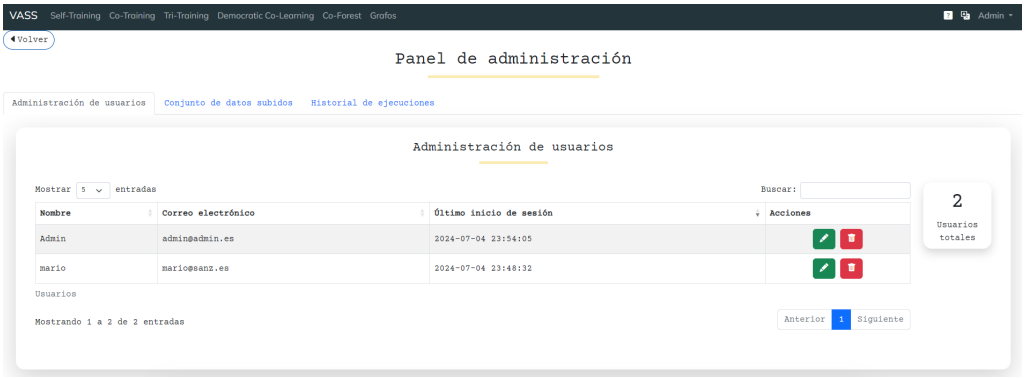

Figura E.46: Administración de usuarios

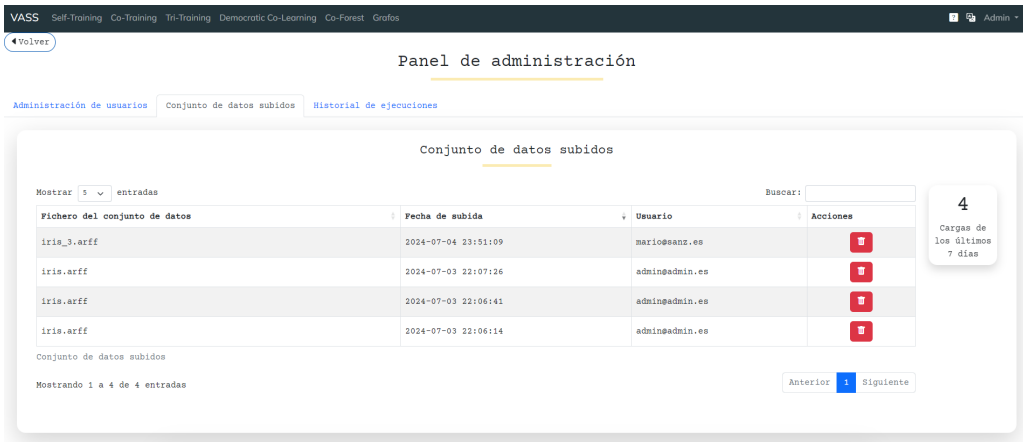

Figura E.47: Conjuntos de datos subidos

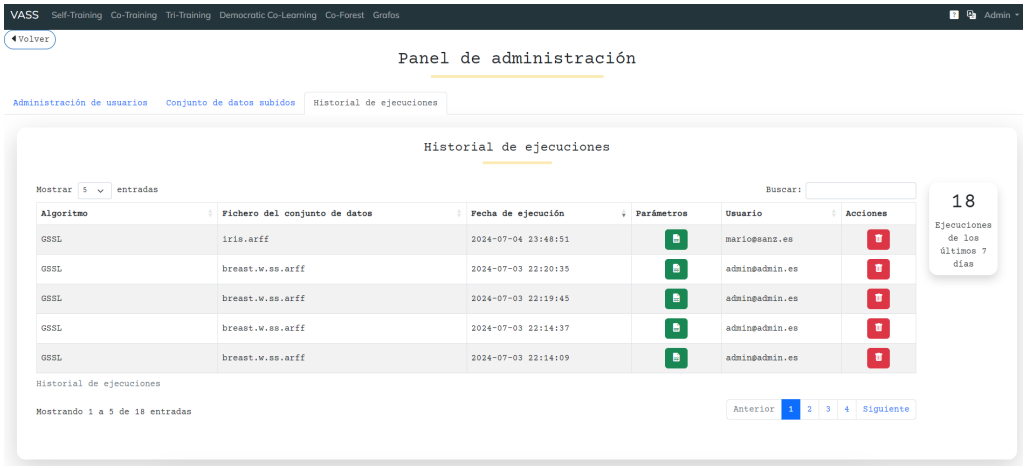

Figura E.48: Historial de ejecuciones

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

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

Si se quiere **editar** un usuario, se debe pulsar el botón  . Esto redirigirá a una pestaña similar a la de la figura E.4 (perfil del usuario) y de hecho, será como adoptar la vista del usuario que se está editando salvo por la inclusión de un indicativo como recordatorio al administrador (ver figura E.49).

Figura E.49: Edición de un perfil ajeno

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

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

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

Si lo que se quiere es **eliminar** un usuario, el proceso es el mismo que se ha visto para el resto de eliminaciones. Se debe pulsar en el botón  y se pedirá confirmación del proceso.

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

El proyecto realizado se centra en el desarrollo de una herramienta web destinada a la docencia de algoritmos de aprendizaje semisupervisados. A lo largo del desarrollo del proyecto, se han abordado varios aspectos relacionados con la sostenibilidad, tanto en el ámbito social como ambiental y económico. El documento https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf define las competencias en sostenibilidad que deben ser integradas en la formación universitaria, las cuales se comentan a continuación.

Competencia en la Contextualización Crítica del Conocimiento

Durante el desarrollo del proyecto, se ha realizado un esfuerzo significativo en comprender cómo la tecnología desarrollada puede influir en la educación y la sociedad. La herramienta no solo facilita el aprendizaje de algoritmos complejos, sino que también promueve el acceso equitativo a recursos educativos avanzados, lo cual es esencial para un desarrollo social justo y equitativo. Esta competencia se ha reflejado en la adaptación de la plataforma para ser accesible a usuarios con diferentes niveles de habilidades y conocimientos.

Competencia en la Utilización Sostenible de Recursos

El diseño y la implementación de la versión 2.0 del visualizador se ha llevado a cabo con un enfoque en la eficiencia de los recursos tecnológicos. Se ha utilizado un servidor que proporciona los servicios mínimos de la aplicación, sin exceder de recursos. Además, se ha priorizado el uso de software libre y herramientas de código abierto, reduciendo la dependencia de tecnologías propietarias y fomentando un entorno de desarrollo más sostenible.

Competencia en la Participación en Procesos Comunitarios

El proyecto fomenta la participación y colaboración dentro de la comunidad educativa. Se ha distribuido bajo una licencia MIT, lo que permite a otros educadores y desarrolladores utilizar y modificar el código fuente de la herramienta. Además, se ha promovido la retroalimentación y la contribución de la comunidad a través de la publicación del código en un repositorio público.

Competencia en la Aplicación de Principios Éticos

La implementación del visualizador de algoritmos se ha llevado a cabo siguiendo estrictos principios éticos. Se ha asegurado que los datos utilizados para el desarrollo y pruebas del sistema sean anónimos y se respete la privacidad de los usuarios.

Bibliografía

[1] David Martínez Acha. Herramienta docente para la visualización en web de algoritmos de aprendizaje semi-supervisado. *Universidad de Burgos*, 2023.

[2] Lilian Berton, Thiago de Paulo Faleiros, Alan Valejo, Jorge Valverde-Rebaza, and Alneu de Andrade Lopes. Rgcli: Robust graph that considers labeled instances for semi-supervised learning. *Neurocomputing*, 226:238–248, 2017.

[3] Expansión. Jornada laboral. https://www.expansion.com/diccionario-juridico/jornada-laboral.html, 2024. [Internet; descargado 17-junio-2024].

[4] Patricia Hernando Fernández. Aprendizaje semisupervisado y ciberseguridad: detección automática de ataques en sistemas de recomendación y phising. *Universidad de Burgos*, 2023.

[5] indeed. Sueldo de programador/a junior en españa. https://es.indeed.com/career/programador-junior/salaries, 2024. [Internet; descargado 17-junio-2024].

[6] indeed. Sueldo de programador/a junior en españa. https://es.indeed.com/career/profesor-universitario/salaries?from=top_sb, 2024. [Internet; descargado 17-junio-2024].

[7] Ming Li and Zhi-Hua Zhou. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE transactions on systems, man, and cybernetics - Part A: Systems and Humans*, 37(6):1088–1098, 2007.

[8] Microsoft. Precios de máquinas virtuales windows. https://azure.microsoft.com/es-mx/pricing/details/virtual-machines/windows/, 2024. [Internet; descargado 3-julio-2024].

[9] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[10] Seguridad Social. Tipos de cotización. https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/10721/10957/9932/4315Âž, 2024. [Internet; descargado 17-junio-2024].

[11] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8174–8194, 2023.

[12] Jesper E. van Engelen and Holger H. Hoos. A survey on semi supervised learning. *Machine Learning*, 109:373–400, 2020.

[13] Guido van Rossum, Barry Warsaw, and Alyssa Coghlan. Pep 8 – style guide for python code. https://peps.python.org/pep-0008/, 2013. [Internet; descargado 10-abril-2024].

[14] Wikipedia. Licencia mit — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Licencia_MIT&oldid=159348811, 2024. [Internet; descargado 10-abril-2024].

[15] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.