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 — 17 de junio de 2024
Tutor: Álvar Arnaiz González

Índice general

Índice de figuras

Índice de tablas

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 [9] 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 [8]. 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. 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. Tambien 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 (descartado por ser de pago), 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 [8].
- 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 [14], 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.
- 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.

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.

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 un servidor con vida útil de 3 años y

Activo	**Coste (€)**	**Coste amortizado (€)**
Ordenador portátil	829	207.25
Monitor ayuda	105.99	26.5
Servidor	X	X
Total	X	X

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).

Activo	**Coste (€)**	**Coste amortizado (€)**
Microsoft 365	69	51.75
Github Copilot	228	171
Dominio web	X	X
Total	X	X

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

Viabilidad legal

[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.

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 y*

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-1.1 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 `ACSV` 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 Descarga de conjunto de datos**: el sistema debe permitir la descarga de estos ficheros a los que se puede acceder y que la aplicación ya contiene.
    - **RF-2.3 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 Visualización de resumen del fichero de datos**: la aplicación debe mostrar al usuario un resumen del conjunto de datos que se ha cargado o se va a utilizar para la ejecución del algoritmo.

- **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-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 a implementación del código**: la aplicación debe permitir acceder a la implementación de los algoritmos como recurso por si se quiere utilizar en el equipo del usuario.
- **RF-6.3 Acceso al manual de usuario**.

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 (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...*), 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 **??** se han pintado de verde aquellos que se han modificado, de rojo los nuevos y de negro los que ya existían.

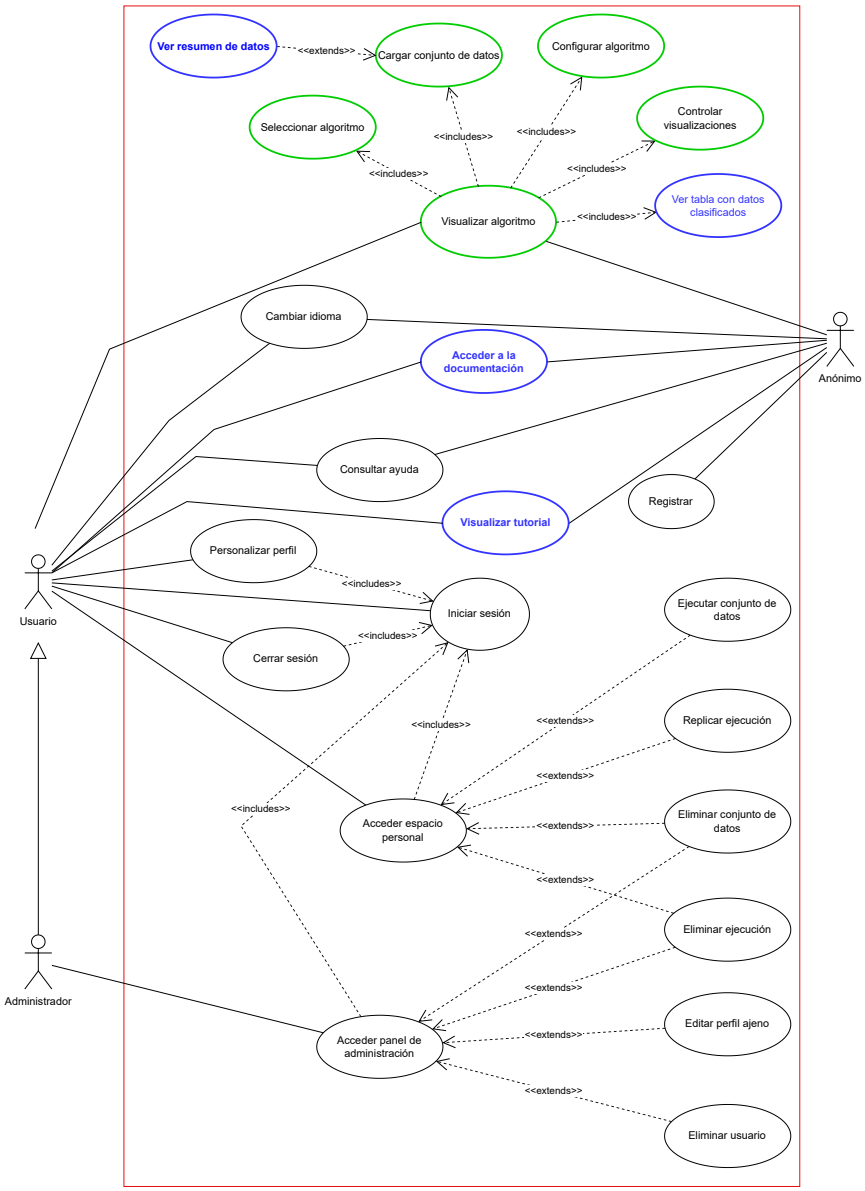

Figura B.1: Diagrama de casos de uso. En verde los modificados, en azul los nuevos y en negro los que ya existían

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 do-

cumentación de https://github.com/dmacha27/TFG-SemiSupervisado/tree/main/doc.

CU-1	**Ejemplo de caso de uso**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-xx, RF-xx
Descripción	La descripción del CU
Precondición	Precondiciones (podría haber más de una)
Acciones	1. Pasos del CU 2. Pasos del CU (añadir tantos como sean necesarios)
Postcondición	Postcondiciones (podría haber más de una)
Excepciones	Excepciones
Importancia	Alta o Media o Baja...

Tabla B.1: CU-1 Nombre del caso de uso.

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

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.

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

[9] Mario Sanz Pérez. Tfg-semi-supervised-learning. https://github.com/msp1015/TFG-Semi-Supervised-Learning, 2023. Github.

[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] 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.