



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Estudio de métodos de
selección de instancias en
aprendizaje Semi-Supervisado
y aplicación web de MLaaS
Documentación Técnica



IS-SSL DNX

Presentado por Daniel Puente Ramírez
en Universidad de Burgos — 5 de junio
de 2022
Tutor: Dr. Álvar Arnaiz González

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	vi
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Usuarios participantes	2
A.3. Planificación temporal	2
A.4. Estudio de viabilidad	40
Apéndice B Especificación de Requisitos	51
B.1. Introducción	51
B.2. Objetivos generales	52
B.3. Usuarios del <i>software</i>	53
B.4. Factores de riesgo	54
B.5. Catálogo de requisitos	55
B.6. Especificación de requisitos	60
Apéndice C Especificación de diseño	73
C.1. Introducción	73
C.2. UBUMLaaS	73
C.3. IS-SSL	83
Apéndice D Documentación técnica de programación	85
D.1. Introducción	85

D.2. UBUMLaaS	85
D.3. IS-SSL	97
Apéndice E Documentación de usuario	111
E.1. Introducción	111
E.2. UBUMLaaS	111
E.3. IS-SSL	132
Bibliografía	139

Índice de figuras

A.1.	Metodología <i>scrum</i> [12].	3
A.2.	<i>Burndown Chart Sprint 1.</i>	8
A.3.	<i>Burndown Chart Sprint 2.</i>	10
A.4.	<i>Burndown Chart Sprint 3.</i>	11
A.5.	<i>Burndown Chart Sprint 4.</i>	13
A.6.	<i>Burndown Chart Sprint 5.</i>	14
A.7.	<i>Burndown Chart Sprint 6.</i>	15
A.8.	<i>Burndown Chart Sprint 7.</i>	17
A.9.	<i>Burndown Chart Sprint 8.</i>	18
A.10.	<i>Burndown Chart Sprint 9.</i>	20
A.11.	<i>Burndown Chart Sprint 10.</i>	21
A.12.	<i>Burndown Chart Sprint 11.</i>	23
A.13.	<i>Burndown Chart Sprint 12.</i>	25
A.14.	<i>Burndown Chart Sprint 13.</i>	27
A.15.	<i>Burndown Chart Sprint 14.</i>	28
A.16.	<i>Burndown Chart Sprint 15.</i>	29
A.17.	<i>Burndown Chart Sprint 16.</i>	30
A.18.	<i>Burndown Chart Sprint 17.</i>	32
A.19.	<i>Burndown Chart Sprint 18.</i>	33
A.20.	<i>Burndown Chart Sprint 19.</i>	35
A.21.	<i>Burndown Chart Sprint 20.</i>	36
A.22.	<i>Burndown Chart Sprint 21.</i>	38
A.23.	Distribución de las horas repartidas por tareas.	39
A.24.	Distribución de las horas repartidas por tareas.	39
A.25.	Orden de permisividad/restricción de licencias.	47
A.26.	Licencia GNU v3 para UBUMlaaS.	48
A.27.	Licencia BSD 3 Clause para IS-SSL.	48

B.1. Diagrama de casos de uso.	60
C.1. Diagrama entidad relación.	75
C.2. Diagrama relacional.	76
C.3. Diagrama de secuencia de la monitorización en tiempo real. . .	79
C.4. Diagrama de secuencia de las estadísticas generales de la aplicación. .	79
C.5. Diagrama de secuencia de la creación de un nuevo experimento por parte del usuario.	80
C.6. Diagrama de secuencia de la ejecución de un nuevo experimento. .	81
C.7. Arquitectura cliente-servidor.	82
D.1. Codacy.	93
D.2. SonarCloud.	94
D.3. Travis-CI.	95
D.4. Codacy.	103
D.5. SonarCloud.	104
D.6. Prueba base para algoritmos de selección de instancias.	105
D.7. Prueba base para algoritmos de aprendizaje semi-supervisado. .	106
D.8. Prueba base para algoritmos de aprendizaje semi-supervisado. .	106
E.1. Página de inicio.	113
E.2. Página de registro.	113
E.3. Página de inicio de sesión.	115
E.4. Página de recuperar contraseña.	115
E.5. Índice principal de UBUMLaaS.	116
E.6. Vista de crear experimento.	116
E.7. Formulario de crear experimento relleno.	118
E.8. Perfil del usuario.	119
E.9. Visualización de resultados.	120
E.10. Vista de antes de predecir.	121
E.11. Vista de después de predecir.	122
E.12. Estadísticas de usuario.	124
E.13. Formulario de edición de los datos de un usuario.	125
E.14. Formulario para cambiar la contraseña de un usuario.	126
E.15. Vista principal de administrador.	127
E.16. Vista de <i>Analytics Dashboard</i>	128
E.17. Vista de administración de usuarios.	129
E.18. Vista de <i>Live System Monitor</i>	130
E.19. Vista de la biblioteca de algoritmos de selección de instancias en PyPI.	133

E.20. Vista de la biblioteca de algoritmos de aprendizaje semi-supervisado en PyPI.	133
E.21. Instalación de la biblioteca de selección de instancias.	134
E.22. Instalación de la biblioteca de semi-supervisado.	134

Índice de tablas

A.1.	Horas dedicadas al proyecto por tareas.	38
A.2.	Costes de <i>hardware</i>	41
A.3.	Costes de <i>software</i>	41
A.4.	Costes de personal.	42
A.5.	Otros costes.	43
A.6.	Costes totales.	43
A.7.	Opciones para obtener beneficio con UBUMLaaS.	44
A.8.	Simulación para recuperar la inversión con UBUMLaaS.	44
A.9.	Bibliotecas utilizadas y sus versiones.	46
A.10.	Bibliotecas utilizadas y sus versiones.	46
A.11.	Bibliotecas utilizadas y sus versiones.	49
B.1.	CU-1 Consultar Experimentos.	61
B.2.	CU-1.1 Consultar Experimento.	62
B.3.	CU-1.1.1 Predecir Nuevas Instancias.	63
B.4.	CU-1.1.2 Reutilizar Experimento.	64
B.5.	CU-1.2 Consultar Registros en la Base de Datos.	65
B.6.	CU-2 Crear Experimento.	66
B.7.	CU-3 Modificar Usuario.	67
B.8.	CU-4 Registro de Usuario.	68
B.9.	CU-5 Administrar Usuarios.	69
B.10.	CU-6 Consultar Analíticas de Uso.	70
B.11.	CU-7 Monitorización del Sistema en Tiempo Real.	71
B.12.	CU-7.1 Monitor del Sistema.	71
D.1.	Bibliotecas utilizadas por UBUMLaaS y sus versiones.	89

D.2. Bibliotecas utilizadas por IS-SSL y sus versiones. Se muestran en negrita únicamente las bibliotecas necesarias para hacer uso de las bibliotecas tal y como están disponibles en PIP, las demás son utilizadas en procesos de experimentación y visualización de resultados.	99
D.3. Comparación de resultados ACC de los algoritmos de selección de instancias, el clasificador base es <i>3-NN</i>	108
D.4. Comparación de resultados ACC de los algoritmos de selección de instancias, el clasificador base es <i>Decision Tree</i>	109
D.5. Comparación de resultados ACC de los algoritmos de aprendizaje semi-supervisado.	110

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este anexo se tratará el plan de proyecto, es la base sobre la que se crea el proyecto *software*. Desde el punto de vista de la temporalidad y la viabilidad. Siendo una parte fundamental del proyecto ya que permitirá visualizar el escenario en el que se desarrollará, permitiendo hacer una alineación estratégica de todos los elementos que se deben completar para finalizarlo correctamente.

Desde el punto de vista de la planificación temporal, el proyecto sigue la metodología ágil *Scrum*. Permitiendo definir cada uno de los objetivos que se desean alcanzar, los elementos que los componen y su respectiva prioridad.

Scrum, de manera muy resumida, trabaja con un *product backlog*, es una lista de prioridades en función del valor de cada tarea. Cuando comienza un *sprint*, se empieza a trabajar en las tareas que se encuentren en el *sprint backlog*, estas han sido extraídas del *product backlog*. En el caso de este proyecto se realiza una reunión de planificación, *sprint planning*, cada dos semanas inicialmente y posteriormente cada semana.

Para el control y seguimiento se utiliza una herramienta externa, *Zenhub*, la cual permite la definición de las tareas, el seguimiento de cada una de ellas en función de la planificación póker, seguimiento de cada *sprint*, el versionado, etc.

Seguidamente se realizará un estudio de la viabilidad del proyecto, tanto a nivel económico como legal.

A.2. Usuarios participantes

En la fase de análisis han participado diversos usuarios, entre los que se han repartido los principales «papeles».

- Dr. Álvar Arnaiz González, tutor del proyecto, ha sido partícipe de múltiples papeles a lo largo de esta fase:
 - *Cliente*. Descripción de las funcionalidades deseadas de la aplicación y el comportamiento que debe de tener.
 - *Técnico*. Aportando conocimientos acerca de las técnicas de selección de instancias y su relación con la minería de datos. Junto con ello ha compartido sus conocimientos en el uso de bibliotecas tales como Weka, Scikit-Learn, o el lenguaje de marcas LATEX. Así como el aporte de grandes cantidades de documentación en forma de *papers* o documentación web.
- Multitud de compañeros del grado han aportado sus experiencias a la hora de tratar con aplicaciones de este tipo, comentando sus principales dificultades que encuentran habitualmente y lo que esperarían encontrarse en una nueva aplicación. Lo que permite hacer un diseño de la interfaz más intuitivo en función de lo que el usuario espera encontrar sin perder funcionalidades.
- *Analista*. El alumno ha realizado el análisis (valga la redundancia) y descripción del problema planteado por el cliente y realización del diseño de la solución propuesta.

A.3. Planificación temporal

SCRUM

Scrum es un marco de trabajo que permite el trabajo colaborativo en equipos. Permite que los equipos que trabajan en proyectos con esta metodología se organicen por sí mismos, siendo ellos los que deciden cómo afrontar los problemas que van surgiendo.

Según [18], el modelo *Scrum* se basa en tres componentes principales: roles, procesos y artefactos. El *Scrum Master* es el puesto asumido por el director o gerente del proyecto, o en algunos casos el líder del equipo. Esta figura representa los valores y principios por los que se rige la metodología de *scrum*, manteniendo los valores y buenas prácticas, así como resolviendo

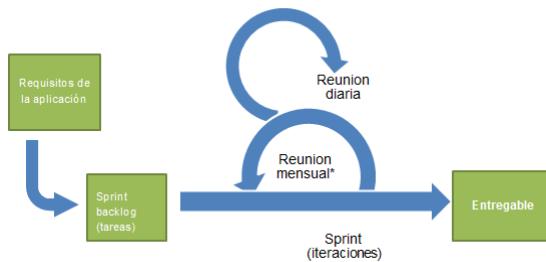


Figura A.1: Metodología *scrum* [12].

los impedimentos que vayan surgiendo a lo largo del desarrollo del proyecto. Habitualmente los equipos están compuestos por entre cinco y diez personas que trabajan en el proyecto a tiempo completo. Siendo este equipo independiente y flexible en cuanto a jerarquía interna, no siendo representado el papel del «jefe» dentro de este por la misma persona siempre. Esto genera que el papel cambie en función de las necesidades del propio proyecto, la configuración del equipo cambia únicamente entre iteraciones, o *sprints*, no dentro de los mismos.

Sprints

Los *sprints* son períodos breves de **tiempo fijo** en el que el equipo trabaja para completar una cantidad de trabajo pre-establecida. Si bien muchas guías asocian los *sprints* a la metodología ágil, asociando la metodología ágil y la metodología seguida en *scrum* como si fueran lo mismo, cuando no lo son. La metodología ágil constituye una serie de principios, y la metodología *scrum* es un marco de trabajo con la única finalidad de conseguir resultados.

A pesar de las similitudes los *sprints* poseen un objetivo subyacente, entregar con frecuencia *software* de trabajo.

Sprint meetings

Dentro de la metodología *scrum* existen diferentes reuniones que favorecen la agilidad del proyecto y que todo el mundo sepa lo que tiene que hacer en cada momento.

- ***Sprint planning meeting.*** Esta reunión puede tener una duración de hasta de un día completo de trabajo. En ella deben de estar presentes todas las partes del proyecto, *i.e.* el *Scrum Master*, el equipo de

desarrollo, y el *product owner*. Poseen dos partes, en la primera de ellas se define el *product backlog*, requerimientos del proyecto y se definen los objetivos para el *sprint* que comienza, *i.e.* lo que se espera «construir» o completar en el *sprint*. En la segunda parte de la reunión se trabaja en el *sprint backlog*, las tareas que se van a seguir en el *sprint* para completar el objetivo de éste.

- **Daily meeting.** Debido a que los requerimientos del proyecto no se pueden variar durante la vida de un *sprint*, existen las reuniones diarias que son organizadas por el *Scrum Master* en las que se comenta el trabajo del día previo, lo que se espera de ese día y qué está retrasando o impidiendo a un individuo el proseguir con sus tareas, esta reunión no debe tener una duración de más de quince minutos y se debe realizar «de pie». No es una reunión para ver quién retrasa el proyecto sino para ayudar a quién lo necesite entre todos los miembros del equipo y permitir esa agilidad.
- **Sprint review meeting.** Reunión fijada al final de cada *sprint* en la cual se hace una puesta en conocimiento de lo que se ha realizado en ese *sprint*, siempre que se pueda se hará una demostración funcional en lugar de una presentación al *product owner*. Esta reunión tiene un carácter informal.

Artifacts

Uno de los componentes más importantes de cara a la metodología *scrum* son los artefactos, o *artifacts* por su nombre en inglés. Éstos incluyen el *product backlog*, el *sprint backlog* y los *burn down charts*.

- **Product backlog.** Lista de trabajo ordenada por las prioridades para el equipo de desarrollo. Es generada a partir de las reuniones de planificación de los *sprints*, contiene los requisitos. Se encuentra actualizado y clasificado en función de la periodicidad asignada a las tareas, pudiendo ser de corto o largo plazo. Aquellas tareas que se deban resolver a corto plazo deberán estar perfectamente descritas antes de asignarlas esta periodicidad, implicando que se han diseñado las historias de usuario completas así como el equipo de desarrollo ha establecido las estimaciones correspondientes. Los elementos a largo plazo pueden ser abstractos u opacos, conviene que estén estimados en la medida de lo posible para poder tener en cuenta el tiempo que llevará desarrollarla.

Los propietarios del producto dictan la prioridad de los elementos de trabajo en el *product backlog*, mientras que el equipo de desarrollo dicta la velocidad a la que se trabaja el *backlog* [32].

La estimación es una parte muy importante ya que es lo que permitirá al equipo de desarrollo mantener el ánimo y el trabajo al ritmo deseado. La estimación es realizada en la *sprint planning meeting*, en la que se estima para cada tarea/producto del *product backlog*. No se busca tener un resultado exacto del tiempo que va a llevar al equipo completar esa tarea, sino es una previsión. Para realizar correctamente la estimación se debe tener en cuenta el tamaño y la categoría de la tarea, los puntos de historia que se le van a asignar, así como el número de horas y días que van a ser necesarias para completar la tarea.

- **Sprint backlog.** Lista de tareas extraídas del *product backlog* que se han acordado desarrollarse a lo largo de un *sprint*. Este *backlog* es seleccionado por el propio equipo de desarrollo, para ello seleccionan una tarea del *product backlog* y se divide en tareas de menor tamaño y abordables. Aquellas tareas de menor tamaño que el equipo no haya sido capaz de desarrollar previo a la finalización del *sprint* quedarán almacenadas para próximos *sprints* en el *sprint backlog*.

Actores, roles y responsabilidades

Dentro de un equipo que sigue la metodología *scrum* encontramos diferentes actores, como ya se ha comentado el equipo de desarrollo suele estar compuesto por entre cinco y diez personas, además del *Scrum Master* y el *Product Owner* [33].

- **Product Owner.** Encargado de optimizar y maximizar el valor del producto, es la persona encargada de gestionar las prioridades del *product backlog*. Una de sus principales tareas es la de ser intermediario con los *stakeholders*, partes interesadas, del proyecto; junto con recoger los requerimientos de los clientes. Es habitual que esta figura sea representante del negocio, con lo que aumenta su valor.

Para cada *sprint* debe de marcar el objetivo de éste de manera clara y acordada con el equipo de desarrollo, lo cual hará que el producto vaya incrementando constantemente su valor. Para que todo fluya como debe, esta figura debe tener el «poder» de tomar decisiones que afecten al producto.

- **Scrum Master.** Figura con dos responsabilidades, gestionar el proceso *scrum* y ayudar a eliminar impedimentos que puedan afectar a la entrega del producto.
 1. Gestionar el proceso *scrum*. Su función es asegurarse de que el proceso se lleva a cabo correctamente, facilitando la ejecución de éste y sus mecánicas. Consiguiendo que la metodología sea una fuente de generación de valor.
 2. Eliminar impedimentos. Eliminar los problemas que vayan surgiendo a lo largo de los *sprints* con el fin de mantener el ritmo de trabajo dentro de los equipos de desarrollo para poder entregar valor, manteniendo la integridad de la metodología.
- **Equipo de desarrollo.** Formado por entre cinco y diez personas encargadas del desarrollo del producto, organizados de forma autónoma para conseguir entregar las tareas del *product backlog* asignadas al *sprint* correspondiente. Para que funcione correctamente la metodología todos los integrantes deben de conocer su rol dentro del equipo, internamente se pueden gestionar como el equipo considere, pero de cara «hacia fuera» son un equipo con una responsabilidad.

Planificación por *sprints*

La organización temporal del proyecto se ha organizado siguiendo los estándares de la metodología *scrum*, *i.e.* usando *sprints*.

Inicialmente la *sprint planning meeting* es realizada cada dos semanas, debido a una falta de costumbre de trabajo con esta metodología se combina junto con la *sprint review meeting*, de forma que en una sola reunión se comenta tanto lo que se ha hecho como lo que está por realizarse en el siguiente *sprint*.

La velocidad de desarrollo del proyecto es una incógnita, debido a la no existencia de referencias previas del equipo de desarrollo del proyecto, en proyectos de ésta índole. Por lo tanto, la duración de los *sprints* puede que se vea ajustada a lo largo de la vida del proyecto.

No se utilizan *daily meetings* puesto que a pesar de que se invierte una media de tres a cinco horas diarias en el desarrollo, no es considerada necesaria. Si bien en caso de problemas se acuerda una reunión para el día siguiente con el fin de mantener la agilidad y no retrasar el proyecto.

Sprint 0: Lights out and away we go!

El *sprint* con el que comienza el desarrollo de este proyecto no ha seguido la metodología *scrum*, puesto que se formuló desde un punto de vista de toma de contacto inicial con el trabajo de investigación y todo lo que ello conlleva.

Los objetivos definidos han sido:

1. Lectura de *papers* relacionados con el ámbito de la inteligencia artificial.
En concreto *SSL density peaks* [38], *Co-Training* [14], *Tri-Training* [41] y *Democratic Co-Learning* [40].

El tiempo empleado en la lectura y asimilación de estos conceptos ha sido de catorce horas, es la primera vez que se leen *papers* o artículos científicos completos procurando asimilar todos los conceptos de éstos. Se ha desarrollado entre el 27/10/2021 y 05/11/2021.

Sprint 1: Chad

- ***Planning meeting***

Objetivos del primer *sprint*:



Figura A.2: *Burndown Chart Sprint 1*.

1. Lectura del API de *scikit-learn*. Comprensión del funcionamiento de los transformadores y estimadores enfocado desde el punto de su programación.
2. Lectura de los *papers*: *On issues instance selection* [30], *Comparison of instances selection algorithms I. algorithms survey* [27] y *Comparison of instance selection algorithms II. Results and comments* [22].
3. Implementación de las técnicas de reducción del conjunto de entrenamiento, basados en *k-NN*.

■ **Marcas temporales**

El *sprint* se desarrolla entre 08/11/2021 y 19/11/2021. Han sido dedicadas al desarrollo del proyecto treinta horas.

■ ***Burndown chart***

Durante este *sprint* el trabajo inicial comenzó ligeramente retrasado, motivos en el apartado *sprint review meeting*, por lo tanto podemos observar en la Figura A.2 como el trabajo completado dista del ideal o proyectado para este *sprint*.

En el *sprint backlog* habían sido incluidos todos los algoritmos a programar, es por ello que indica que se ha completado aproximadamente la mitad del trabajo.

- **Sprint review meeting**

El trabajo en este primer *sprint* ha salido adelante correctamente. Al ser el primer *sprint* ha habido un pequeño error de configuración del repositorio junto con la herramienta ZenHub, de ahí que en el *burndown chart* de esta semana, Figura A.2, aparezca como que la primera semana del *sprint* no ha habido trabajo completado.

La adaptación a la metodología ágil ha resultado un poco compleja.

Sprint 2: Holleyman

- **Planning meeting**

Objetivos del segundo *sprint*:

1. Finalizar implementación de los algoritmos basados en técnicas de reducción del conjunto de entrenamiento.
2. Añadir la documentación correspondiente a los algoritmos implementados.
3. Comprobar el rendimiento de los algoritmos implementados respecto a los resultados de una ejecución similar con el software *Weka*.

- **Marcas temporales**

El *sprint* se desarrolla entre 20/11/2021 y 04/12/2021.

- **Burndown chart**

El trabajo realizado a lo largo de este *sprint* ya ha sido adecuado a la metodología *scrum*, obteniendo un *burndown chart*, Figura A.3, con más sentido que la que se había obtenido en el *Sprint 1*.

El equipo de desarrollo se sigue habituando poco a poco a la metodología de trabajo y en este *sprint* se ha trabajo por debajo del «ideal» para el proyecto.

- **Sprint review meeting**

A lo largo de este *sprint* se descubrió un problema en la forma de identificar los *k-NN* en el algoritmo *Condensed Nearest Neighbor, CNN* [23], retrasando el trabajo cuatro horas, entre identificación y recodificación. Este error se descubrió mientras se investigaba otro error, en este caso el algoritmo *Iterative Case Filtering, ICF* [17] terminaba en error buscando los *k-NN* de las últimas instancias.

La implementación de los algoritmos *Reduced Nearest Neighbor, RNN* [21] y *Modified Selective Subset, MSS* [13] ha sido relativamente asequible

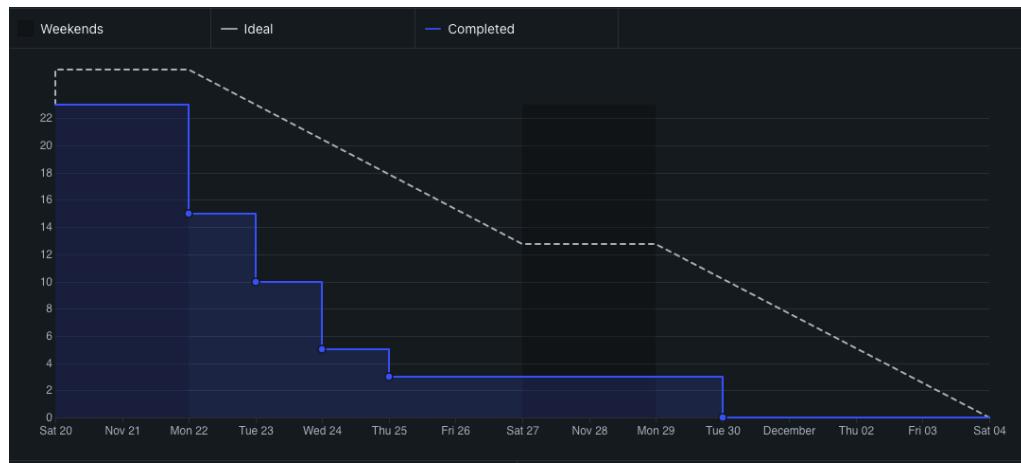


Figura A.3: *Burndown Chart Sprint 2.*

una vez se comprendía el algoritmo en cuestión así como su funcionamiento (entradas, procesado, salidas...).

Sprint 3: Manion

■ ***Planning meeting***

Objetivos del tercer *sprint*:

1. Comenzar la documentación del proyecto.
 - Comenzar la memoria por el marco teórico.
 - Comenzar los anexos por la planificación temporal.
- Se va a realizar en \LaTeX .
2. Aprender lo básico de \LaTeX lo más rápido posible para poder trabajar con él.
 3. Buscar la precisión de los algoritmos implementados con conjuntos etiquetados de [1 %, 5 %, 10 %, 20 %, 40 %, 60 %, 80 %, 100 %] del conjunto total. En búsqueda de las asintotas donde ya no mejora la clasificación.
 4. Validación de los algoritmos de selección de instancias con **Weka** y **KNN**.

Figura A.4: *Burndown Chart Sprint 3.*

- **Marcas temporales**

El *sprint* se desarrolla entre el 06/12/2021 y el 17/12/2021. Han sido dedicadas al desarrollo del proyecto 45 horas.

- **Burndown chart**

El trabajo en este tercer *sprint* ha sido realizado a un ritmo constante y con una dedicación en número de horas un poco mayor a los anteriores, como se puede ver en el *Burndown report*, ver Figura A.4, el número de *story points* de este *sprint* era de 39 y a pesar de que algunas tareas llevaron más tiempo del inicialmente planificado, otras resultaron ser totalmente lo contrario, mucho más rápidas de realizar.

- **Sprint review meeting**

Este *sprint* ha sido un poco más grande en cuanto a horas de trabajo invertidas ya que el tiempo del equipo de desarrollo así lo ha permitido. A su vez se han detectado *bugs* en la codificación de algoritmos como ICF (se arreglará en el siguiente *sprint*) el cual después de comparar sus resultados contra los expresados por Weka con 9 conjuntos de datos no considerados como de juguete, la codificación del proyecto obtiene soluciones 20 % inferiores; el resto de los algoritmos implementados están en el rango de $\pm 2\%$. A su vez también se han descubierto limitaciones de otros algoritmos como es el caso de ENN cuando tiene pocas muestras y un número elevado de clases diferentes.

Se ha proseguido con la redacción de la memoria del trabajo, finalizando la primera parte de conceptos teóricos y comenzando la explicación teórica de los algoritmos que se van a implementar.

Sprint 4: The Seven

■ ***Planning meeting***

Objetivos del cuarto *sprint*:

1. Revisar y corregir la codificación del algoritmo *Iterative Case Filtering, ICF*.
2. Formatear las métricas de rendimiento recogidas durante el *sprint* anterior.

■ ***Marcas temporales***

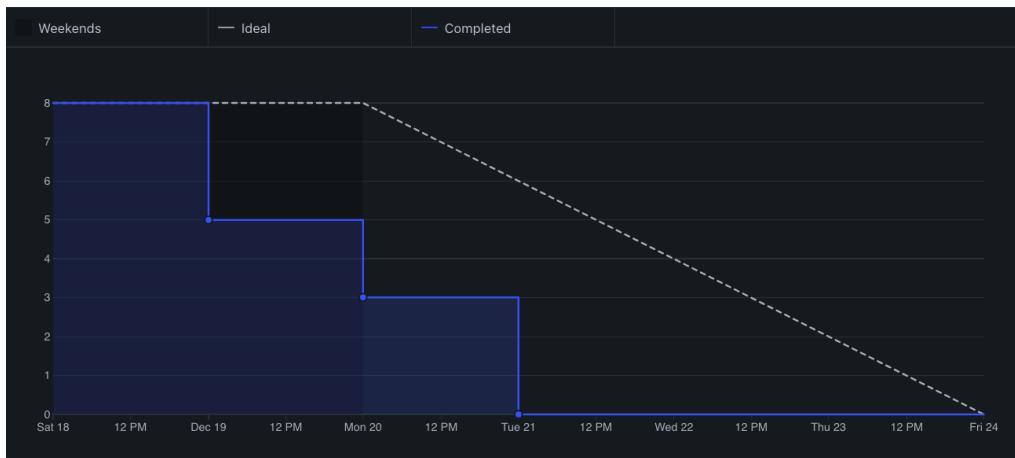
El *sprint* se desarrolla entre el 18/12/2021 y el 23/12/2021. Han sido dedicadas al desarrollo del proyecto 21 horas. Este *sprint* posee una duración más corta con el fin de ajustar las reuniones a las festividades propias de la Navidad.

■ ***Burndown chart***

En el *Burndown report* asociado a este *sprint*, ver Figura A.5, se aprecia como el trabajo ha sido finalizado en unas marcas temporales muy por delante de lo «ideal». Esto se debe a que el equipo de desarrolló comenzó a realizar el trabajo el sábado 18 de diciembre, en lugar de esperar al lunes 20, bajo la presunción de que el trabajo asignado iba a ser mayor.

■ ***Sprint review meeting***

A pesar de la corta duración del *sprint* para poder organizar el siguiente *sprint* antes de las festividades navideñas, el trabajo realizado ha sido correcto e importante, debido a que para poder seguir trabajando en otros algoritmos de selección de instancias o de aprendizaje semi-supervisado, lo anterior debe de quedar correctamente hecho. Es por ello, que prácticamente se le ha dedicado un *sprint* entero a arreglar el algoritmo *Iterative Case Filtering, ICF*, ya que con él y a falta de implementar DROP3, ya tendríamos todos los algoritmos de selección de instancias correctamente implementados.

Figura A.5: *Burndown Chart Sprint 4.*

Sprint 5: Murph

- ***Planning meeting***

Objetivos del quinto *sprint*:

1. Codificación de los algoritmos de aprendizaje semi-supervisado: Co-Training [14], Tri-Training [41], Democratic Co-Learning [40]. Y el algoritmo de selección de instancias DROP3 [37].
2. Implementar los algoritmos anteriores como clases para poder ser utilizados con métodos *fit* y *predict*.
3. Escribir la documentación de la memoria referente a los anteriores algoritmos.
4. Escribir la planificación temporal relativa a los *sprints* 4 y 5.
5. Añadir leyenda a la figuras generadas con *self-training* en función de un % de datos etiquetados.

Se espera que sea un *sprint* muy productivo debido a las fechas en las que se realiza y la mayor disponibilidad del equipo de desarrollo.

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 24/12/2021 y el 10/01/2022. Tiene una duración igual a las festividades navideñas.



Figura A.6: *Burndown Chart Sprint 5.*

■ **Burndown chart**

En este *sprint*, y como vemos en la Figura A.6, referente al correspondiente *Burndown report*; se ha realizado una gran cantidad de trabajo, habiendo siendo completados 77 puntos de historia, una cantidad muy superior a anteriores *sprints*, esto es debido en gran parte a las fechas en las que nos encontramos, ya que el número de horas que se han podido invertir en el desarrollo del proyecto ha sido muy superior a lo que venían siendo habituales. En total han sido utilizadas cerca de 110 horas de trabajo, siendo repartidas en los 17 días que ha durado el *sprint* y con una media de horas de trabajo de 6.5 horas diarias.

■ **Sprint review meeting**

Todo el trabajo que se ha realizado en este *sprint* podría haber sido realizado seguramente en tres cuartas partes del tiempo real invertido, pero debido al tiempo de lectura de los artículos de donde se extraían los algoritmos, así como su correcta comprensión, codificación y posterior resolución de problemas asociados a *bugs* que se van descubriendo «sobre la marcha», ha sido un *sprint* largo y en algunos momentos agotador.

A falta de realizar las correspondientes pruebas de validación de los algoritmos implementados, para comprobar que son correctas las implementaciones, ya se encontrarían finalizados todos los algoritmos de selección de instancias.

Figura A.7: *Burndown Chart Sprint 6.*

Sprint 6: Bert

- ***Planning meeting***

Objetivos del sexto *sprint*:

1. Verificación de la correcta implementación del algoritmo de selección de instancias DROP3. Se realizará como se ha venido trabajando anteriormente, contra los resultados propuestos para la misma parametrización, por Weka.
2. Verificación de la correcta implementación de los algoritmos de aprendizaje semi-supervisado: *Co-Training*, *Tri-Training* y *Democratic Co-Learning*.
3. Comenzar a escribir las secciones de «Técnicas y herramientas» y «Trabajos relacionados».

- ***Marcas temporales***

Este es un *sprint* relativamente corto, puesto que es de verificación de que el trabajo realizado hasta el momento es correcto, antes de pasar a otro «bloque» de trabajo. Comienza el martes 11/01/2022, y finaliza el lunes 17/01/2022.

- ***Burndown chart***

Tal y como se aprecia en la Figura A.7 referente al sexto *sprint*, el

ritmo de trabajo ha sido constante a lo largo de la primera semana, torciéndose al final del *sprint* debido a la complejidad sobrellevada de aprender la biblioteca *Flask* y sus dependencias. Es por ello que dos *issues* se cerraron un día más tarde de la planificación. El número de horas aproximado que se han invertido han sido de 50h, permitido en gran medida con que todavía no hay clases del segundo cuatrimestre.

- ***Sprint review meeting***

El trabajo realizado en este *sprint* ha ido de acuerdo con lo que se comentó en la anterior reunión. Si bien ha sido un *sprint* más enfocado a «cerrar» la parte de trabajo que se llevaba realizada para poder comenzar con el mejor pie posible la segunda etapa.

Un punto de inflexión realizado en este *sprint* ha sido la refactorización de gran parte del repositorio, dejándolo en un formato de paquetes.

Sprint 7: Felix The Cat

- ***Planning meeting***

Objetivos del séptimo *sprint*:

1. Modificar el algoritmo ENN para poder utilizarlo con Semi-Supervisado según el método de borrado de instancias.
2. Preparar *scripts* para la experimentación y posterior visualización de hipótesis.
3. Modificar la memoria en función de los comentarios de Alvar.
4. Añadir a los trabajos relacionados UBUMLaaS. Aunque es parte del propio Trabajo de Fin de Grado, no deja de ser una herramienta de MLaaS.
5. Añadir *Self Training* a UBUMLaaS.

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 25/01/2022 y el 02/02/2022. Es un *sprint* muy rápido de preparación para poder comenzar con la parte de UBUMLaaS.

- ***Burndown chart***

El *Burndown* de este *sprint* representa un ritmo de trabajo «adelantado» a la velocidad óptima, esto se debe a que como en el *sprint* anterior no se cerraron todas las *issues* previa la finalización de este, pero sí fueron cerradas previo el inicio de este nuevo *sprint*, el gráfico

Figura A.8: *Burndown Chart Sprint 7.*

queda por debajo siempre. El número de horas invertido ha rondado las 35h. Un *sprint* de menor tamaño.

- ***Sprint review meeting***

Este *sprint* si bien es como el anterior muy corto, y se ajusta a la temporalización del proyecto, ha tenido una carga de trabajo un poco más alta de lo esperado, esto se debe a que la integración de nuevos algoritmos a UBUMLaaS parecía en un primer momento muy directo, pero se han requerido hacer modificaciones con las que no se contaba en un primer momento.

Sprint 8: Jason

- ***Planning meeting***

Objetivos del octavo *sprint*:

1. Crear una nueva selección en «Nuevo Experimento» en UBUM-LaaS para los algoritmos de aprendizaje Semi-Supervisado.
2. Integrar los algoritmos implementados de Semi-Supervisado en la plataforma UBUMLaaS.
3. Comenzar a traducir parte de la interfaz como parte de un trabajo paralelo. (Puede que la versión final soporte varios idiomas, decisión de diseño aún por tomar.)



Figura A.9: *Burndown Chart Sprint 8.*

4. Crear los rankings con Python de las experimentaciones realizadas, principalmente de 3-NN sin borrado.
5. Hacer un *refactor* general al proyecto. Inicialmente tenía una estructura de carpetas, se desea una estructura de paquetes.
6. Hacer el proyecto accesible desde PIP¹.

■ **Marcas temporales**

Este *sprint* se desarrolla entre el 03/02/2022 y el 08/02/2022. Nos encontramos ante otro *sprint* ya con la nueva dinámica de trabajo, de duración aproximada a una semana.

■ **Burndown chart**

Tal y como se aprecia en la Figura A.9 se aprecia que el ritmo de trabajo en este *sprint* ha sido muy escalonado, el número de *issues* no ha sido muy elevado, pero la complejidad de estas sí que lo ha sido, es por ello que se planificó 90 puntos de historia. Seguidamente podemos apreciar como entre el cuatro y el siete de febrero, coincidiendo con el fin de semana, no ha habido trabajo finalizado, se debe a unas mini-vacaciones que se tomó el equipo de desarrollo.

■ **Sprint review meeting**

Con el *sprint* finalizado se ha visto como lo que parecía una planifica-

¹Sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

ción para una o dos semanas, ha quedado resuelta dentro del propio *sprint*. El equipo de desarrollo comienza a familiarizarse con el *backend* de UBUMLaas propiciando un desarrollo más eficaz de las tareas que se van encomiando.

Destacar que no se finalizan todas las tareas en tiempo, sino que se finaliza una en la noche del martes ocho, ya casi de madrugada, entrando técnicamente en el siguiente *sprint*.

Sprint 9: Lumberjack 20

- ***Planning meeting***

Objetivos del noveno *sprint*:

1. Continuar con la traducción del *frontend* en los «tiempos muertos», aún no se ha decidido si finalmente pasará a producción o no.
2. Crear un nuevo conjunto de gráficas y relanzar experimentaciones con SVC como referencia. El método de eliminación de instancias con etiqueta conocida queda descartado, únicamente se trabajará para la experimentación con la aproximación que no las elimina.
3. Crear los nuevos rankings basados en la precisión.
4. Comprobar la implementación de los algoritmos *Co-Training*, *Tri-Training* y *Democratic Co-Learning* contra los implementados por Jose Luis Garrido Labrador (Investigador del grupo ADMIRABLE).
5. Añadir a UBUMLaas los filtros implementados en los anteriores *sprints*.

- **Marcas temporales**

Este *sprint* se desarrolla entre el 09/02/2022 y el 16/02/2022.

- ***Burndown chart***

Este *sprint* tiene una duración de una semana, tal y como se desea (aproximadamente) que sean a partir de febrero. A este *sprint* se le asignó una gran carga en cuanto a puntos de historia se refiere con un total de 102, las horas invertidas por el equipo de desarrollo no han llegado a las 55, hay una desviación de los puntos de historia y las horas invertidas, se comentará más adelante.

Quedando el trabajo finalizado un par de días antes de la fecha de finalización del *sprint*, dando al equipo de desarrollo tiempo para planear futuras tareas y aproximaciones a problemas conocidos.



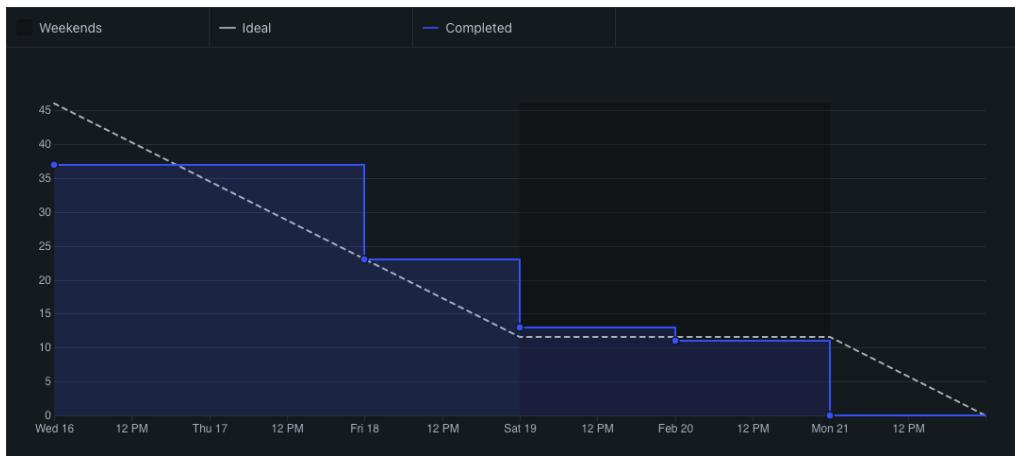
Figura A.10: *Burndown Chart Sprint 9.*

- ***Sprint review meeting***

En este *sprint* se planificó «tirando a lo alto» en los puntos de historia, se debe a que se requería comprobar la implementación de los tres algoritmos de aprendizaje Semi-Supervisado, y en caso de que alguno (o todos) tuviera una implementación incorrecta, realizar los ajustes pertinentes para que fuera correcta. Debido a la experiencia del equipo de desarrollo un mes atrás con el filtro ICF, el cual tuvo que ser recodificado y revisado en más de una ocasión debido a su inconsistencia, se aventuró un futuro similar con éstos algoritmos ya que son más grandes y con una complejidad superior. La realidad en este caso superó las expectativas del equipo de desarrollo, cuando los tres algoritmos tuvieron una desviación menor al 1% en comparación con los de referencia. (En futuros *sprints* se ha propuesto ser más críticos con la asignación de puntos de historia para no tener diferencias de este calibre).

Con todo y con ello, la implementación de los filtros en UBUMlaaS incurrió en múltiples modificaciones a la estructura base de la propia plataforma, pero con un resultado satisfactorio.

Los *rankings* creados no han convencido en estructura y formato, es por ello que en siguiente *sprint* tendrán que ser repetidos.

Figura A.11: *Burndown Chart Sprint 10.*

Sprint 10: Jerry

- ***Planning meeting***

Objetivos del décimo *sprint*:

1. Rehacer las gráficas de rankings en la experimentación.
2. Comenzar con la parte de administración de UBUMLaaS.
 - a) Crear una nueva interfaz que de soporte a esta nueva funcionalidad que va a poseer la aplicación.
 - b) Integrar nuevos campos en el registro de usuarios, tales como su país de origen y el uso deseado que se le va a dar a la aplicación.
 - c) Crear una interfaz de administración de usuarios (añadir usuarios, activarlos, hacerlos administradores o eliminarlos).
 - d) Crear una primera interfaz básica de *dashboard analytics* del sistema.

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 16/02/2022 y el 21/02/2022.

- ***Burndown chart***

Este *sprint* ha sido más ajustado el número de horas invertidas en el desarrollo de las tareas marcadas en comparación con los puntos de historia. Se han marcado un total de 46 puntos de historia y se

han invertido 35 horas de trabajo. Siguiendo un poco más la tónica de otros *sprints*. En los primeros días, tal y como se aprecia en la Figura A.11, sí que hubo *commits* pero no se cerraron tareas debido a que se trabajó en «paralelo» sobre varias *issues* a la vez, ya que toda la parte de crear la interfaz de administración y las páginas que la iban a comenzar a formar parte de la misma, se encuentran fuertemente inter-relacionadas.

- ***Sprint review meeting***

El trabajo realizado durante este *sprint* ha sido más duro que el de *sprints* anteriores. Esto se debe a la poca experiencia del equipo de desarrollo con aplicaciones que poseen un *frontend*, el uso de JavaScript, jQuery, AJAX,... es algo que hasta la fecha no se había utilizado en gran medida y ahora es con lo que más se está trabajando, entonces ha requerido de un esfuerzo extra.

La parte de administración de UBUMlaaS ha sido creada con una base más moderna, sencilla y clara. Siguiendo el esquema de colores de la Universidad de Burgos. Es por ello que ahora mismo parecen dos aplicaciones diferentes, (la parte de administración en comparación con la parte de funcionalidad de MLaaS propiamente dicha).

Durante la realización del *sprint* fueron surgiendo pequeños *bugs* en la interfaz gráfica que se fueron solventando, todos ellos originados por descuidos (debido a la falta de experiencia) del propio equipo de desarrollo con el uso de las nuevas librerías.

Sprint 11: Nutts

- ***Planning meeting***

Objetivos del undécimo *sprint*:

1. Con [29] se desea comprobar con 16 de los 18 conjuntos de datos utilizados en sus experimentos los resultados esperados para comprobar si merece la pena continuar la línea de investigación con el enfoque inicial.
2. Montar un servidor con Jenkins, se desea incorporar a UBUMlaaS y a la biblioteca *IS_SSL* dentro de CI/CD ²

²Método de distribución de aplicaciones a los clientes con una cierta frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Se trata de una solución para los problemas que se pueden generar en la integración del código nuevo en producción.

Figura A.12: *Burndown Chart Sprint 11.*

3. Añadir al *dashboard analytics* gráficos de carta con el número de experimentos de cada tipo que se han ejecutado. Así como los tiempos de uso de cada algoritmo.
4. Crear una pantalla de carga para el *dashboard* de forma que la recuperación de datos sea asíncrona.
5. Permitir al usuario añadir más datos personales dentro de su perfil (Institución, redes sociales, ...)
6. Realizar una nueva página de usuarios con la nueva distribución.
7. Permitir al usuario ver sus propias estadísticas de uso.
8. Pantalla tipo *dashboard* con el estado en directo del sistema.

■ **Marcas temporales**

Este *sprint* se desarrolla entre el 22/02/2022 y el 01/03/2022.

■ **Burndown chart**

Con una duración de algo más de una semana, se han planificado un total de 58 puntos de historia para estos días. Las horas de trabajo han sido cercanas a las 45. La sensación del equipo de desarrollo después de haber finalizado el *sprint* es de un trabajo a ritmo constante finalizando tarea tras tarea, esta sensación se puede comprobar como en efecto ha sido así en la Figura A.12.

- ***Sprint review meeting***

En este *sprint* no se han podido terminar todas las tareas, si bien en el servidor local en el que corre UBUMLaaS se ha podido desplegar Jenkins, el *pipeline* para que funcione correctamente no se ha podido terminar. Igual se buscan otras alternativas que además den soporte a elementos como las *badges* de GitHub y visualización de si pasan o no los tests en los propios *commits*.

Las principales pantallas de administración van quedando mejor con cada *sprint*, más retoques se las van haciendo y el equipo de desarrollo poco a poco comienza a sentirse más cómodo trabajando con lenguajes de marcas como es HTML, o de programación como JavaScript.

El número de horas invertidas en las que no se está programando como tal, sino que se requieren de aprendizaje previo a poder escribir código y hacer la tarea X que toque, sigue siendo alto en esta parte del proyecto.

Sprint 12: DVB

- ***Planning meeting***

Objetivos del duodécimo *sprint*:

1. Se ha decidido dejar «en pausa» la traducción de UBUMLaaS a idiomas como el castellano o el francés. No se descarta retomarlo en un futuro o que sean líneas de trabajo futuro.
2. Con la parte de administración ya más avanzada y con una cohesión mayor, se ha tomado la decisión de dar un «lavado de cara» a toda la aplicación, esto implica rehacer **todas** las páginas del *frontend* con el fin de que se adapten a la nueva guía de estilo de la aplicación.
3. Realizar pequeños ajustes a ejes de gráficos.
4. Decidir e implementar una forma de toma de datos en tiempo real del sistema anfitrión para en posteriores *sprints* visualizar esa información.
5. Implementación del algoritmo de aprendizaje Semi-Supervisado basado en picos de densidad, ver [39].

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 01/03/2022 y el 08/03/2022.

Figura A.13: *Burndown Chart Sprint 12.*

- ***Burndown chart***

El trabajo realizado en este *sprint* tal y como en la Figura A.13 se aprecia, ha sido superior a los anteriores, con un total de 83 puntos de historia y cerca de 45 horas invertidas. En esta ocasión el trabajo ha vuelto a ser planificado «por lo alto» debido a la suposición de complejidad de implementación del algoritmo de aprendizaje Semi-Supervisado.

- ***Sprint review meeting***

En este *sprint* el equipo de desarrollo ha tenido la sensación de que no «llegaba» a todo lo planificado, las reuniones llegan a un punto en el cual se comenta trabajo, queda apuntado, y se intenta meter todo en tiempo y forma. Generando un cierto agobio en algunas situaciones que han impedido continuar con el trabajo al ritmo deseado.

En líneas generales se puede afirmar que el *frontend* ha sido rehecho entero, se han reutilizado formatos o formularios existentes por facilidad de uso a todos aquellos usuarios que ya la conocieran, pero a nivel de código prácticamente es nueva. Con un estilo mucho más moderno, fino y elegante.

La integración de CI/CD finalmente ha quedado hecha con elementos *cloud*, entre ellos se encuentran Travis-CI, Codebeat, SonarCloud y Codacy. Debido a que se ha rehecho toda la interfaz web de la aplicación, los tests existentes no pasan, es por ello que se tendrán

que rehacer poco a poco, aunque no es uno de los elementos de mayor prioridad por el momento.

Sprint 13: Kutschbach

■ *Planning meeting*

Objetivos del decimotercero *sprint*:

1. Implementación del algoritmo de aprendizaje Semi-Supervisado basado en picos de densidad con filtrado, ver [29].
2. Mejora inicial de la calidad del código.
3. Panel *dashboard* de visualización de estado del sistema en tiempo real.
4. Dar soporte a que el usuario pueda cambiar su foto de perfil.
5. Realizar algunas pruebas de estrés para detectar puntos de rotura de la interfaz.
6. Comenzar a escribir los Requisitos.
7. Comenzar a escribir dentro del Diseño, el diagrama de casos de uso.
8. Añadir a los aspectos relevantes los métodos que se están haciendo así como los cambios en la interfaz.
9. Revisar comentarios hechos por Alvar en la memoria.

■ *Marcas temporales*

Este *sprint* se desarrolla entre el 08/03/2022 y el 15/03/2022.

■ *Burndown chart*

Tal y como se aprecia en la Figura A.14, en este *sprint* el ritmo de trabajo ha sido muy constante, algunas tareas fueron asignadas con puntos de historia más bajos de lo que deberían de haber sido y así quedan reflejados en los días diez y once. El total de puntos de historia ha sido de 48 con un total de 30 horas invertidas.

■ *Sprint review meeting*

El trabajo realizado en este *sprint* ha sido satisfactorio a pesar de que no se han podido terminar todas las tareas abiertas a tiempo, esto ha sido debido a un pequeño bajón en la motivación del equipo de desarrollo junto con otras actividades de la vida universitaria.

En lo referido al proyecto, han surgido múltiples reconsideraciones del

Figura A.14: *Burndown Chart Sprint 13.*

diseño de la interfaz según se iban recuperando datos y diseñando, por lo que el proceso de trabajo ha tenido un componente creativo en muchas ocasiones, no siendo el principal fuerte del equipo de desarrollo.

Sprint 14: T.U.P.

- ***Planning meeting***

Objetivos del decimocuarto *sprint*:

1. Ultimar detalles de los casos de uso.
2. Hacer diagramas de secuencia de «Nuevo Experimento», «Consultar Analíticas de Uso» y «Monitorización en tiempo real».
3. Realizar la experimentación con picos de densidad y ruido.
4. Implementación de los algoritmos LSSm y LSBo, ver [28].

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 15/03/2022 y el 22/03/2022.

- ***Burndown chart***

En este *sprint* se han invertido cerca de 38h, el equipo de desarrollo cada vez se encuentra más cómodo trabajando en las diferentes tareas que se asignan, y aunque el número de puntos de historia es relativamente elevado, 60, esto es debido al histórico de dificultad de programar determinados algoritmos.



Figura A.15: *Burndown Chart Sprint 14*.

- ***Sprint review meeting***

Junto con lo expuesto anteriormente, se ha comprobado en este *sprint* la implementación de los algoritmos basados en picos de densidad, con la coronada de que no habría algún fallo en su implementación. Para sorpresa del equipo de desarrollo no han sido necesarios cambios a mayores de un par de «fallos de dedo» a la hora de programarlos, lo cual ha permitido una mayor agilidad a la hora de trabajar y reducir el número de horas invertidas.

Sprint 15: Robbie

- ***Planning meeting***

Objetivos del decimoquinto *sprint*:

1. Toma de decisión del formato del diagrama de secuencia «Nuevo Experimento».
2. Introducir en Trabajos Relacionados, una disyunción entre UBUMaaS y el aprendizaje semi-supervisado seguro.
3. Comenzar con la primera etapa de experimentación «seria» que se va a realizar.

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 22/03/2022 y el 29/03/2022.

Figura A.16: *Burndown Chart Sprint 15.*

- ***Burndown chart***

Lo primero a destacar de este *sprint* y tal cual lo refleja la Figura A.16, correspondiente al *Burndown report*; no se han terminado todas las tareas a tiempo. Esto ha sido debido a que faltaba por cerrar un *pull request* el cual estaba pasando una serie de tests.

El número total de puntos de historia asignados al *sprint* ha sido de 45, y se han invertido aproximadamente 35 horas, en esta ocasión el trabajo ha ido acorde a los puntos de historia asignados.

- ***Sprint review meeting***

Con la experimentación lanzada y pudiendo haberse hecho entera, únicamente un sexto de lo que se espera que sea al final, los resultados no parecen ser muy prometedores, pero aún es pronto para saber lo que finalmente va a ser.

Los diagramas de secuencia han llevado mucho más tiempo del inicialmente esperado, esto se debe a la poca experiencia realizando este tipo de actividades y que no son el principal atractivo, por lo que el trabajo en esas partes se ha visto ralentizado.

En general todas las tareas relacionadas con la memoria están requiriendo más tiempo del que *a priori* parece que va a ser necesario. Pero para conseguir un producto de calidad, es lo que se debe hacer.



Figura A.17: *Burndown Chart Sprint 16.*

Sprint 16: Matt 16

- ***Planning meeting***

Objetivos del dieciseisavo *sprint*:

1. Mejora de la calidad del código de los algoritmos de la biblioteca IS-SSL.
2. Remates de los diagramas de secuencia.
3. Añadir *Self-Training* basado en picos de densidad [39] a los Conceptos Teóricos.
4. Escribir el Manual del Programador.
5. Crear ficheros de configuración de entorno para Conda y Pyenv.

- ***Marcas temporales***

Este *sprint* se desarrolla entre el 29/03/2022 y el 08/04/2022.

- ***Burndown chart***

En este *sprint* se ha trabajado principalmente en la memoria y en retoques de código, es por ello que se ha dado una cifra superior de puntos de historia de la media, y con lo visto en el *sprint* anterior, estas tareas están comenzando a llevar más tiempo de que inicialmente se piensa. Se han invertido aproximadamente 37 horas. Los tiempos de trabajo empiezan a ser correctos de forma reiterada con lo planificado.

- **Sprint review meeting**

Con la experimentación a tres sextos realizada, todavía no se ha encontrado un nexo común que nos permita crear hipótesis, por lo que se aprovecharán las vacaciones de Semana Santa para dejar más experimentos en ejecución y rehacer las *scripts* de análisis de resultados.

UBUMLaaS parece que está correcto en todas sus funcionalidades, por lo que ya se podría afirmar que la parte «grande» de modificación está terminada.

Sprint 17: Dae Han

- **Planning meeting**

Objetivos del decimoséptimo *sprint*:

1. Modificar las tablas de versiones con una descripción.
2. Realizar una encuesta para utilizar agentes externos como *beta testers* para UBUMLaaS.
3. Cambiar las *Long Table* de los casos de uso por tablas normales de L^AT_EX.
4. Escribir el anexo de la documentación técnica del programador.
5. Realizar los diagramas de relación.
6. Modificar los algoritmos de visualización de los resultados de la experimentación.

- **Marcas temporales**

Este *sprint* se desarrolla entre el 08/04/2022 y 20/04/2022. Englobando las vacaciones de Semana Santa.

- **Burndown chart**

El trabajo a lo largo de este *sprint* ha sido a ritmo constante, tal y como se aprecia en la Figura A.18, se aprecia como el número de puntos de historia es elevado, esto se debe a que se realizan en total 26 *issues*, del total de 27 planificadas. Son muchas tareas pequeñas que acaban elevando el número de puntos de historia empleados. En total se han invertido alrededor de 45 horas de trabajo a lo largo de todo el *sprint*.

- **Sprint review meeting**

En este *sprint* se sobre-predijo el número de puntos de historia necesarios para resolver un número determinado de *bugs* en UBUMLaaS,



Figura A.18: *Burndown Chart Sprint 17.*

resultando estos más sencillos de resolver que en lo que en un primer momento parecía.

Una tarea que quedó por cubrir fue la escritura de las pruebas que se han realizado en los productos *software* desarrollados, no dando tiempo a finalizarlo en tiempo y forma.

El *sprint* ha estado muy enfocado en mejorar la calidad del código de las bibliotecas de IS-SSL, así como la escritura de la memoria y anexos.

Sprint 18: Desforges

- ***Planning meeting***

Objetivos del decimoctavo *sprint*:

1. Por solapamiento con la asignatura de Minería de Datos, se va a introducir el algoritmo RESSEL [20].
2. Finalizar el anexo «Especificación de Diseño».
3. Finalizar el anexo «Documentación de usuario».
4. Realización del estudio de la viabilidad legal y económica del proyecto.
5. Actualizar la tabla de *Experiments* en la base de datos de UBUMLaas para que los algoritmos de Scikit-Learn sean compatibles con la versión 0.24.

Figura A.19: *Burndown Chart Sprint 18.*

- **Marcas temporales**

Este *sprint* se desarrolla entre el 20/04/2022 y 02/05/2022.

- **Burndown chart**

Tal y como se aprecia en la Figura A.19, en este *sprint* el ritmo de trabajo no ha sido continuo, por motivos de índole personal la primera semana de trabajo no se pudieron cerrar prácticamente tareas, si bien sí que se pudo avanzar en ellas permitiendo que cuando se dispuso de tiempo para trabajar a ritmo «normal» en ellas, éstas se cerraran con relativa sencillez.

El tiempo total invertido fue de 30 horas aproximadamente. Quedando cerrados 50 puntos de historia.

- **Sprint review meeting**

La calidad del producto que se está obteniendo está resultando satisfactoria para ambas partes, tanto el equipo de desarrollo como para el tutor del proyecto.

Cabe destacar que el producto de documentación final (memoria y anexos) está empezando a ser relativamente grande, no siendo un mal indicador ya que a la fecha de finalización del *sprint* ya se han realizado numerosas revisiones de esta, pero ello no implica que no llame la atención.

Se detectó y solucionó un problema en los algoritmos de *clustering* en UBUMLaaS, gracias a las pruebas realizadas por terceros sobre el

proyecto. Además, se actualizaron los algoritmos de **Scikit-Learn**, ya que su parametrización se había modificado y no permitía su ejecución.

Sprint 19: Rahoi

■ ***Planning meeting***

Objetivos del decimonoveno *sprint*:

1. Mejora de la calidad del código de las bibliotecas **IS-SSL**, principalmente reducir la complejidad cognitiva de los algoritmos de aprendizaje semi-supervisado.
2. Mejora de la documentación del código de las librerías **IS-SSL**.
3. Solución de errores reportados por **SonarCloud** en **UBUMLaaS**.
4. Escritura de las pruebas de **IS-SSL** y **UBUMLaaS** en el anexo de «Documentación técnica de programación».
5. Crear nuevas bases de datos «limpias» para desplegar en producción.
6. Embeber **UBUMLaaS** en un contenedor de **Docker** y dejarlo listo para su despliegue.
7. Crear gráficas de rankings para la experimentación.

■ ***Marcas temporales***

Este *sprint* se desarrolla entre el 03/05/2022 y 10/05/2022.

■ ***Burndown chart***

El trabajo en este *sprint* ha ido al ritmo deseado, si bien en la Figura A.20 se aprecia como el viernes 06 no se cerraron grandes tareas, el trabajo del sábado se vio afectado permitiendo cerrar más de las habituales. Ha sido un *sprint* que ha requerido de bastantes horas «extras» de investigación y aprendizaje para poder cumplimentar todas las *issues* correctamente.

Aproximadamente se han invertido cerca de 41 horas en el desarrollo del proyecto esta semana. Siendo resueltos 73 puntos de historia.

■ ***Sprint review meeting***

La calidad del producto está aumentando considerablemente, y a la vista de los resultados se aprecia la facilidad de mantenimiento. Gracias a las sucesivas iteraciones de mejora de la calidad, cada *sprint* se puede mejorar y mantener el código con más facilidad, permitiendo «hacer más en menos».

Figura A.20: *Burndown Chart Sprint 19.*

Un *bug* que se detectó y paralizó el trabajo fue el descubrimiento de que UBUMLaas había dejado de funcionar, pero únicamente en sus capacidades de experimentación. Fue necesaria casi una jornada entera de trabajo para solucionar el problema. En el proceso de refactorizaciones se hicieron pequeños cambios en las declaraciones de las colas del sistema, anteriormente estando definidas por 4 *workers*, y estos ya no son suficientes para el tamaño de la plataforma, cuando se aumentaron a 16, el problema desapareció.

Las gráficas de rankings no muestran una tendencia, será necesario calcular los rankings medios y realizar *tests* estadísticos sobre estos para poder llegar a conclusiones.

Sprint 20: Klepto

- ***Planning meeting***

Objetivos del vigésimo *sprint*:

1. Crear *tests* estadísticos para los rankings medios.
2. Introducir el trabajo de Triguero de aprendizaje semi-supervisado seguro [35] en los «Trabajos Relacionados» de la memoria.
3. Revisar todas las tablas de la documentación y normalizar, así como hacerlas autocontenidoas si no lo son ya.
4. Añadir a la documentación las validaciones de los algoritmos implementados.



Figura A.21: *Burndown Chart Sprint 20.*

5. Crear y actualizar los repositorios de GitHub, en especial los README.
6. Escribir el «Resumen» e «Introducción» de la memoria.
7. Escribir una primera versión de las «Conclusiones y Líneas futuras de trabajo» de la memoria.

■ **Marcas temporales**

Este *sprint* se desarrolla entre el 10/05/2022 y 16/05/2022.

■ **Burndown chart**

Tal y como se puede comprobar en el *Burndown Report* de este *sprint*, ver Figura A.21, el ritmo de trabajo de este *sprint* ha sido constante. En los primeros días no se pudo cerrar tareas, pero posteriormente el trabajo salió adelante en tiempo y forma. Marcando un remanente de 5 puntos de historia a la finalización del *sprint*, estos puntos de historia eran referentes a *pull requests* que había que verificar antes de poder hacer el *merge* con la rama principal.

Ha sido *sprint* donde no se han invertido muchas horas, siendo esto reflejado en los puntos de historia, aproximadamente se han invertido un total de 16 horas. Una cifra muy por debajo de lo que ha venido siendo la tónica habitual a lo largo de todo el desarrollo del proyecto.

■ **Sprint review meeting**

La memoria y los anexos ya están alcanzando sus estados finales en múltiples capítulos y anexos, respectivamente. El proceso de normalización de pies de figura/tabla, así como el de términos, ha sido

fundamental para otorgar a la memoria un mayor grado de cohesión y coherencia entre los diferentes apartados que posee.

La formalización de los READMEs de los repositorios, así como la creación de la imagen de la librería y por consecuente del proyecto, favorecen el aspecto profesional y serio que se le ha querido dar desde el primer momento.

Finalmente se han añadido las validaciones que se han hecho a todos los algoritmos implementados, así como sus correspondientes tablas de comprobaciones, Tablas D.3, D.4 y D.5. Agrupando en una única sección estos resultados de validación.

Sprint 21: Pheezy

- **Planning meeting**

Objetivos del vigésimo primer *sprint*:

1. Añadir la experimentación realizada a los Aspectos Relevantes de la memoria.
2. Calcular el resumen de horas invertidas por tipo de tareas.
3. Finalizar el estudio económico con cifras.
4. Actualizar la *release* de IS-SSL.
5. Actualizar IS-SSL en PyPI.
6. Crear la primera y única *release* de UBUMlaaS.
7. Actualizar figuras de los anexos (CI/CD, PyPI, ...) con sus últimas versiones.

- **Marcas temporales**

Este *sprint* se desarrolla entre el 17/05/2022 y 23/05/2022.

- **Burndown chart**

Los puntos de historia, ver Figura A.22, en este *sprint* han sido muy pocos, 16 en concreto, siendo reflejados en menos de 14 horas de trabajo. Este *sprint* según la figura ha tenido un gran «escalón», esto se ha debido a que el trabajo ha sido agrupado en tareas épicas y al cierre de éstas es cuando se han cerrado, en grupo. Así mismo refleja que no han sido finalizadas todas las tareas, a tiempo, esto es debido a que el cierre de *pull requests* por algún motivo no lo cuenta bien. Todas las tareas y *PRs* fueron cerradas a tiempo.

Figura A.22: *Burndown Chart Sprint 21.*

- ***Sprint review meeting***

En este *sprint* se ha finalizado la documentación de la memoria, formalizado los repositorios y revisado en la mayor parte la documentación.

Resumen

A continuación se muestra un resumen del tiempo dedicado a cada tipo de tarea. El cálculo ha sido realizado en función de los puntos de historia asignados mediante ZenHub, a razón de 45 minutos por punto de historia.

	<i>Issues</i>	Tiempo (h)	Ptos historia
Bug	32	72,75	97
Code Quality	19	40,5	54
Documentation	82	231	308
Experimentation	15	53,25	71
Feature	87	360,75	481
Research	15	22,5	30
Testing	7	14,25	19
Training	5	29,25	39
Validation	14	15,75	21
Total	276	840	1111

Tabla A.1: Horas dedicadas al proyecto por tareas.

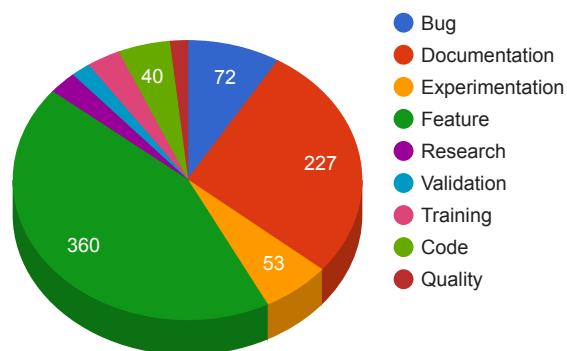


Figura A.23: Distribución de las horas repartidas por tareas.

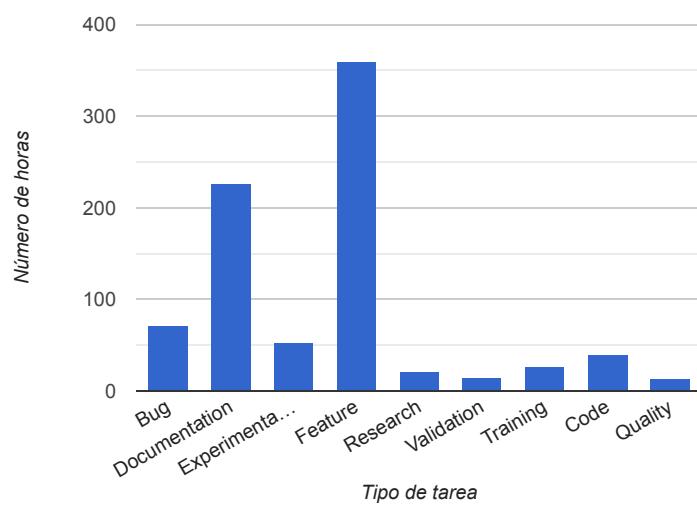


Figura A.24: Distribución de las horas repartidas por tareas.

A.4. Estudio de viabilidad

En esta sección se va a desarrollar el estudio de la viabilidad del proyecto, permitiendo tener una visión global de los beneficios en contraposición al coste que supone el desarrollo del proyecto.

El desarrollo de cualquier producto *software* conlleva una serie de riesgos, entre los que destacan la experiencia del equipo de desarrollo, el tamaño del proyecto, el tiempo del que se dispone para conseguirlo, . . . , influyendo todos ellos en el resultado final.

Viabilidad económica

Lo primero de todo que se va a calcular es la viabilidad económica del proyecto, reportando y analizando los costes/beneficios que habría supuesto el desarrollo del proyecto en el sector empresarial, en España³.

Costes

La realización de un proyecto de esta envergadura lleva asociados una serie de costes fijos y variables, a continuación se desglosan agrupados en *hardware*, *software*, personal, otros.

Costes *hardware*

Para el desarrollo del proyecto es necesario más de un equipo *hardware*, lo primero de todo es un equipo de tipo portátil. Se utilizará un MacBook Pro, con un procesador Inter Core i7 de 4 núcleos a 2.3 GHz, con 16 GB de memoria RAM, atendiendo a un precio de mercado de 2249 €, se tiene en cuenta que la vida útil del equipo se encuentra en torno a los seis años, por lo que para los cálculos se usa la vida media del inmovilizado, es decir, 3 años.

A su vez se necesita un servidor para desplegar la aplicación y trabajar sobre esta, ya que el despliegue local no se considera una forma de trabajo óptima. Es por ello que se hace uso de un equipo con un AMD FX(tm)-4130 Quad-Core a 4.5 GHz, con 64 GB de memoria RAM. El coste aproximado del equipo es de 1500 €. La vida útil estimada es de seis ocho años, se utilizará igual que con el equipo portátil, su vida media de inmovilizado, 4 años.

³Se hace referencia al lugar de desarrollo puesto que se van a hacer referencias económicas en la moneda del país, así como uso de legislación vigente, la cual varía en función del país en el que se encuentre asentada la empresa desarrolladora.

La amortización para cada equipo es diferente, pero el tiempo de uso es el mismo, de noviembre de dos mil veintiuno a junio de dos mil veintidós, ambos incluidos, luego en total ocho meses. Se detallan todos los costes *hardware* en la Tabla A.2.

Concepto	Coste (€)	Coste Amortizado (€)
MacBook Pro	2.249	499,78
Servidor	1.500	250
Total	3.049	749,78

Tabla A.2: Costes de *hardware*.

Costes *software*

Para el desarrollo del proyecto y uso del *software* necesario, es necesaria la adquisición de una serie de licencias, todas ellas se encuentran detalladas con su amortización en la Tabla A.3. Todas las licencias tienen un periodo de validez de un año.

Concepto	Coste (€)	Coste Amortizado (€)
Codacy	170,76	113,84
FileZilla	0	0
GitKraken	59,4	36,27
Codacy	170,76	113,84
Linux Mint 20.3	0	0
MacOs Monterey	0	0
PyCharm	240,79	160,53
Selenium	0	0
SonarCloud	120	80
TExMaker	0	0
Travis-CI	719,86	479,81
Visual Paradigm	93,89	62,59
Visual Studio Code	0	0
Total	1.399,70	933,13

Tabla A.3: Costes de *software*.

Coste de personal

El desarrollo del proyecto ha sido llevado a cabo por un desarrollador y el tutor del proyecto.

- El salario del desarrollador se calcula según [8], siendo el salario medio anual del 20.201 € netos al año.
- El salario del tutor se calcula según [7], siendo el salario medio anual de 22.784 € netos al año. Con una carga de trabajo de dos horas semanales.

La duración total del proyecto es de 31 semanas, por lo tanto, el tutor trabajará un total de 62 horas. Frente a las 840 del desarrollador.

El IRPF es del 24 % y la retribución a la Seguridad Social, calculada tal cual se plantea en [19] por el Ministerio de Inclusión, Seguridad Social y Migraciones; se contribuye con un 31,40 % en total. Estando dividido en:

- 23,60 % de contingencias comunes [16].
- 5,50 % por desempleo de tipo general [16].
- 0,20 % destinado al Fondo de Garantía Salarial [16].
- 0,60 % de formación profesional [16].
- 1,50 % de tipo de cotización por accidentes de trabajo y enfermedades profesionales [15].

Concepto	Desarrollador (€)	Tutor (€)
Salario total neto	1.082,20	81
Retención IRPF	259,73	19,44
Seguridad Social	339,81	25,43
Total salario bruto	1.681,74	125,87
Total 7 meses	11.772,17	881,12

Tabla A.4: Costes de personal.

En la Tabla A.4 se desglosan punto por punto los costes de personal, resultando en:

$$11.772,17\text{€}/desarrollador + 881,12\text{€}/tutor = 12.653,29\text{€}$$

Siendo el coste total en personal de 12.653,29 €.

Otros costes

Costes no agrupables en los anteriores apartados pero a tener en cuenta, Tabla A.5.

Concepto (€)	Coste (€)
Dominio despliegue	32
Logos	45
Memoria impresa	250
Alquiler oficina	1600
Internet	136
Electricidad	140
Agua	51
Calefacción	230
Total	2484

Tabla A.5: Otros costes.

Total de costes

En la Tabla A.6 se puede ver un resumen de todos los costes comentados. Dejando el coste base total del proyecto en 19.585,99 €.

Categoría (€)	Coste (€)
<i>Hardware</i>	3.049
<i>Software</i>	1.399,70
Personal	12.653,29
Otros	2.484
Total	19.585,99

Tabla A.6: Costes totales.

Beneficios

El proyecto está compuesto de dos partes, las cuales se pueden comercializar de forma independiente.

- **IS-SSL.** Las bibliotecas de algoritmos son distribuidas de forma pública, gratuita y sin publicidad, con el fin de aportar un «granito de arena» a la investigación en aprendizaje semi-supervisado y selección de instancias.
- **UBUMLaaS.** En el diseño existente no está planteada su monetización. Si se quisiera obtener un rédito económico de este MLaaS, se podría fácilmente crear niveles de usuarios para acceso, número de

experimentos máximos a lanzar. También se podría plantear como servicio de suscripción. En la Tabla A.7 se plantean algunas posibilidades de comercialización con sus posibles respectivos costes para los usuarios, todas las licencias son anuales, en caso de un usuario quedarse sin tiempo podría comprar más horas de ejecución.

Tipo	Experimentos	Tiempo (h)	Precio (€)	EUR/h
<i>Trial</i>	5	3	0	X
Estudiante	25	20	10	X
Investigador	120	250	40	0.4
Equipo	300	1000	150	0.4
Empresa	Ilimitados	5000	2000	0.25
Otros	Contactar con soporte para obtener presupuesto			

Tabla A.7: Opciones para obtener beneficio con UBUMLaas.

Teniendo en cuenta los anteriores valores, y que la aplicación no tiene «fronteras» ya que el inglés es el idioma oficial de la investigación y la ciencia, se pueden alcanzar usuarios de todo el mundo. En la Tabla A.8 se presenta un ejemplo real y viable de alcanzar la cual permitiría recuperar la inversión del proyecto en el primer año.

Tipo	Número de cuentas	Beneficios (€)
<i>Trial</i>	Ilimitadas	0
Estudiante	4	40
Investigador	15	600
Equipo	40	6.000
Empresa	15	30.000
Otros	No computado	
Total		36.640

Tabla A.8: Simulación para recuperar la inversión con UBUMLaas.

Debido a las funcionalidades ofrecidas y al precio competitivo de la plataforma, no sería difícil encontrar organizaciones que incorporen UBUMLaas en su repertorio de herramientas, maximizando beneficios.

Conclusiones

Analizando los beneficios reportados en contraposición a los costes del desarrollo del proyecto, queda más que demostrada la viabilidad de desarrollo del proyecto. Se tiene en cuenta además que el proyecto una vez desarrollado no tiene necesidad de ser mantenido más allá de añadir nuevos algoritmos. Teniendo como costes fijos el mantenimiento de el(los) servidor(es) en donde esté desplegado.

Viabilidad legal

En esta sección principalmente se va a discutir el tema de licencias, ya que al ser un desarrollo *software* es la temática más importante.

Según [34], una licencia de *software* es un contrato entre la entidad que ha creado y suministrado una aplicación, el código fuente subyacente o un producto relacionado con el mismo, y su usuario final. La licencia es un documento, habitualmente de texto, diseñado para proteger la propiedad intelectual del desarrollador del *software* y para limitar cualquier reclamación contra él que pueda surgir de su uso.

Una licencia proporciona, además, definiciones legalmente vinculantes para la distribución y el uso del *software*. Los derechos del usuario final, como la instalación, las garantías y las responsabilidades, también se encuentran detallados en la licencia.

Software

La licencia más importante es aquella bajo la cual se encuentra el *software* desarrollado por el producto, permitiendo el uso (sobre todo comercial y de distribución que se le puede dar). La principal limitación que se debe tener en cuenta es el orden jerárquico de licencias que se debe respetar. El proyecto utiliza una serie de dependencias externas, las cuales poseen sus propias licencias, nunca pudiendo encontrarse bajo una licencia más permisiva.

A continuación, se exponen las licencias de las dependencias utilizadas tanto por UBUMLaas, Tabla A.9, como por IS-SSL, Tabla A.10. La descripción de cada una se encuentra en las Tablas D.1 y D.2, respectivamente.

Como se aprecia en las Tablas A.9 y A.9, cada proyecto de manera individual va a estar sujeto a diferentes restricciones en cuanto a licencias se refiere, siendo más laxo en IS-SSL y mucho más restrictivo en UBUMLaas.

Dependencia	Versión	Licencia
<code>email-validator</code>	1.1.1	CC0 1.0 Universal
<code>flask</code>	1.1.1	BSD 3 Clause
<code>flask-login</code>	0.4.1	MIT
<code>flask-mail</code>	0.9.1	BSD
<code>flask-migrate</code>	2.5.2	MIT
<code>flask-redis</code>	0.4.0	Blue Oak
<code>flask-sqlalchemy</code>	2.4.0	BSD 3 Clause
<code>flask-wtf</code>	0.14.2	BSD 3 Clause
<code>future</code>	0.16.0	MIT
<code>geopy</code>	2.2.0	MIT
<code>glances</code>	3.2.4.2	GNU GPLv3
<code>imbalanced-learn</code>	0.5.0	MIT
<code>itsdangerous</code>	1.1.0	BSD 3 Clause
<code>liac-arff</code>	2.2.1	MIT
<code>numpy</code>	1.22.3	BSD
<code>pandas</code>	0.25.1	BSD 3 Clause
<code>psutil</code>	5.9.0	BSD
<code>pycountry</code>	22.3.5	GNU LGPLv2
<code>pytest</code>	5.2.1	MIT
<code>python-weka-wrapper3</code>	0.1.7	GNU GPLv3
<code>requests</code>	2.22.0	Apache 2.0
<code>rq</code>	1.1.0	BSD
<code>scikit-learn</code>	0.24	BSD
<code>selenium</code>	3.141.0	Apache 2.0
<code>urllib3</code>	1.25.6	MIT
<code>werkzeug</code>	0.15.6	BSD 3 Clause
<code>whichcraft</code>	0.4.1	BSD

Tabla A.9: Bibliotecas utilizadas y sus versiones.

Biblioteca	Versión	Licencia
<code>numpy</code>	1.20.3	BSD
<code>pandas</code>	1.3.4	BSD 3 Clause
<code>scikit-learn</code>	0.24.2	BSD
<code>scipy</code>	1.7.1	BSD

Tabla A.10: Bibliotecas utilizadas y sus versiones.



Figura A.25: Orden de permisividad/restricción de licencias.

Para la comparación de licencias se utiliza JLA, [36], *Joinup Licensing Assitant*.

En la Figura A.25 se detalla una comparación de las diferentes licencias de las dependencias de cada parte del proyecto, estando más a la derecha aquella con mayor restricción, y a la izquierda la más permisiva. El *software* debe estar licenciado como mínimo, con la más restrictiva.

- **UBUMlaaS.** Estará licenciado bajo GNU v3. Figura A.26.

Siendo una licencia perfecta para los objetivos de comercialización planteados. El *software* se distribuye como *open source* permitiendo que individuos lo usen de forma privada, pero en caso de que se quiera utilizar de cualquier manera de forma comercial deberá ser hecho *open source*, permitiendo al equipo de desarrollo comprobar los cambios realizados, de forma que se puedan añadir al proyecto «base» u original.

- **IS-SSL.** Estará licenciado bajo BSD Cláusula 3. Figura A.27. El objetivo de las bibliotecas que conforman IS-SSL es que puedan ser ampliadas en cualquier momento por cualquier desarrollador, ya sea mediante *pull requests* al proyecto o mediante sus propias implementaciones a través de *forks*. El objetivo del proyecto no es obtener un beneficio económico mediante licencias de uso, por lo tanto, BSD 3 se ajusta a las necesidades y requerimientos del proyecto.

Documentación

La documentación del proyecto también se encuentra licenciada. En este caso no tiene sentido utilizar licencias similares a las anteriores puesto que no es un desarrollo *software*, siendo más comunes y correctas licencias de tipo

 dpr1005/UBUMLaaS is licensed under the
GNU General Public License v3.0

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	① License and copyright notice
✓ Modification	✗ Warranty	① State changes
✓ Distribution		① Disclose source
✓ Patent use		① Same license
✓ Private use		

Figura A.26: Licencia GNU v3 para UBUMLaaS.

 dpr1005/Semisupervised-learning-and-instance-selection-methods is licensed under the
BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	① License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

Figura A.27: Licencia BSD 3 Clause para IS-SSL.

Creative Commons. Se debe tener en cuenta el mismo principio anterior, todo aquel material de terceros que haya sido utilizado para la composición de la documentación, estará sujeto a licencias, no pudiendo ser distribuido con una licencia más permisiva que la que originalmente su autor lo distribuyó.

En la documentación únicamente se han utilizado imágenes de terceros (unas pocas), estado todas ellas licenciadas bajo dominio público (CC0 [1]), permitiendo utilizar cualquier licencia superior de *Creative Commons* para licenciar la documentación.

Se ha decidido utilizar como licencia: *Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0) [2]. En la Tabla A.11 se resume lo que establece, así como acciones permitidas y prohibidas.

Libre de	Bajo los términos
	Atribución
Compartir - Copiar y redistribuir	No Comercial
	Sin Derivadas

Tabla A.11: Bibliotecas utilizadas y sus versiones.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este anexo recoge las necesidades funcionales que deberán ser soportadas por el sistema que va a ser desarrollado. Con el fin de obtener una buena documentación se deben identificar y describir los requisitos que debe el sistema satisfacer, pero sin entrar en cómo los va a resolver.

Hoy en día, no existe una autoridad que indique cómo se deben de realizar las especificaciones de requisitos *software*, SRS. La comunidad se encuentra dividida entre «la vieja escuela» siguiendo guías de buenas prácticas (IEEE 830-1998 [24] ó 12207-2-2020 [25]), en contraposición con el *Agile Manifesto*, donde no se hace una especificación formal de toda la aplicación, sino que cada 2-4 semanas se revisa y «rehace» en función de las necesidades del cliente.

Se va a realizar una combinación de ambos, por un lado, se trabaja a lo largo del proyecto con una planificación ágil, y por otro se va a tener como referencia una especificación de requisitos que va a seguir la guía de buenas prácticas IEEE 830-1998. Ésta última recoge los siguientes puntos como referencias a una buena especificación de requisitos *software* [26].

- **Correcto.** Será correcto si, y sólo si, cada requisito declarado se encuentra en el *software* entregado.
- **Inequívoco.** Será inequívoco si, y solo si, cada requisito declarado tiene sólo una interpretación. Cada característica de la última versión del producto se deberá describir con un único término.

- **Completo.** Será completo si, y sólo si, incluye:
 1. Los requisitos están relacionados a la funcionalidad, el desarrollo, las restricciones del diseño, los atributos y las interfaces externas. En particular debe reconocerse cualquier requisito externo impuesta por una especificación del sistema y debe tratarse.
 2. La definición de las respuestas del *software* a todos los posibles datos de la entrada del sistema y a toda clase de situaciones.
 3. Tener todas las etiquetas llenas y referencias a todas las figuras, tablas, diagramas en el SRS y definición de todas las condiciones y unidades de medida.
- **Consistente.** Si un SRS «choca» con algún documento de nivel superior (*i.e.* una especificación de requisitos de sistema), entonces no será consistente.
- **Comprobable.** Será comprobable si, y sólo si, cada requisito declarado es comprobable. A su vez un requisito será comprobable si, y sólo si, allí existe un proceso rentable finito con que una persona o la máquina puede verificar que el producto del *software* reúne el requisito. Cualquier requisito ambiguo no es comprobable.
- **Modificable.** Será modificable si, y sólo si, su estructura y estilo son tales que puede hacerse cualquier cambio a los requisitos fácilmente, completamente y de forma consistente mientras conserva la estructura y estilo.
- **Identifiable.** Será identificable si el origen de cada uno de los requisitos está claro y facilita de igual manera las referencias de cada requisito de desarrollo futuro o documentación del mismo.

B.2. Objetivos generales

Los objetivos del proyecto se pueden separar en dos ramas.

1. Realización de un estudio de los métodos de selección de instancias más relevantes en la literatura y su aportación en problemas de aprendizaje Semi-Supervisado. Como producto final se desean tener dos bibliotecas con los principales algoritmos de selección de instancias y una de los principales algoritmos de aprendizaje Semi-Supervisado.

2. Integración de las bibliotecas anteriormente expuestas en la plataforma de MLaaS de la Universidad de Burgos (UBUMLaaS).
3. Rediseño completo de UBUMLaaS, modernización de la interfaz gráfica, de forma que sea más intuitivo su uso.
4. Nuevas funcionalidades para el usuario.
5. Administración integral del sistema a cargo del nuevo rol de administrador.

En la biblioteca referida a los algoritmos de filtrado más comunes se implementarán algoritmos clásicos de la literatura como son CNN, RNN, ICF, ... Mientras que la biblioteca de algoritmos clásicos de Semi Supervisado contendrá *Co-Training*, *Tri-Training*, etc. Estando estructuradas en forma de clases accesibles mediante importación clásica de paquetes. Deben de ser fácilmente escalables, posterior a la finalización del proyecto deben poder ampliarse sin añadir complejidad.

Las interfaces a diseñar se requieren que sean intuitivas, fáciles de entender y utilizar. Deberán de ser transparentes al usuario, impidiendo que este conozca la lógica de diseño de la aplicación, así como los posibles fallos internos que se puedan producir por acciones del sistema, del usuario, o de terceros.

B.3. Usuarios del *software*

Cualquier persona podrá hacer uso de la aplicación UBUMLaaS, siendo únicamente necesarios una serie de datos básicos para su registro dentro de ella.

Dentro de la aplicación se encuentra el usuario base y una generalización del mismo en forma de administrador.

- **Usuario.** El usuario será el modelo base, en la forma de una persona la cual tendrá las capacidades de: crear experimentos y todas las funcionalidades asociadas con los mismos. Así como editar sus datos personales, añadir nuevos, quitar, etc. Y conocer sus estadísticas de uso de la última semana.
- **Administrador.** Actor generalizado de usuario. Tiene todas las funcionalidades propias del usuario, pero además posee acceso a toda la

parte de administración de la aplicación. En esta nueva parte posee acceso a la monitorización del sistema en tiempo real, a las estadísticas del mismo en cuanto a uso respecta, administración de todos los usuarios, etc.

Un usuario es creado por una persona ajena que quiere registrarse en la aplicación, o por un administrador. Pero un administrador sólo puede ser «creado» (elevación de usuario a administrador) por otro administrador, y lo mismo para el caso contrario, pasar de administrador a usuario.

B.4. Factores de riesgo

En esta sección se va a realizar un análisis de las ‘principales dificultades que se pueden encontrar a lo largo del desarrollo del proyecto *software*. Mediante una identificación preventiva se podrá poner remedio a éstas de una manera más eficiente e impedir «que vayan a más».

Se identifican los siguientes factores de riesgo:

1. **Desconocimiento teórico.** Se posee una cantidad muy limitada de conocimiento en la materia en la que el proyecto transcurre. El proyecto tiene un enfoque fuertemente relacionado con la minería de datos, un área hasta ahora inexplorada. El proyecto ya en su base más pura va a suponer un reto en el día a día.
2. **Documentación a utilizar.** Hasta ahora nunca se ha tratado con *papers* o artículos científicos, mucho menos su lectura y comprensión, análisis y posterior implementación de los algoritmos propuestos. Puede suponer retrasos sin previo aviso un *paper* con una alta complejidad, bien por la condensación de información, bien por la encapsulación de información, o simplemente por los conocimientos que se requieren para entender el documento.
3. **Experiencia modificando un proyecto *software*.** La experiencia personal dictamina que la modificación de proyectos que han sido iniciados por terceros (como se trabaja en la industria) conlleva una etapa de adaptación la cual no suele ser linear, sino exponencial, en función de la complejidad de la aplicación que se desea asimilar.
4. **Mínima experiencia con algunos lenguajes/bibliotecas.** El proyecto requiere del uso de lenguajes de programación como JavaScript,

o lenguajes de marcas como son HTML, CSS, L^AT_EX... o bibliotecas como Flask o Vue. Con las que no se tiene prácticamente experiencia real de uso. Supondrá un esfuerzo extra e impedirá que determinadas tareas sean tan cortas como deberían serlo.

5. **Existencia del usuario final.** Se desconoce el usuario final de la aplicación, por lo que no se podrán realizar talleres, esto motivará a que el proyecto se creará como se cree que el usuario lo esperaría, pero sin su aprobación.
6. **Motivación del equipo de desarrollo.** En un proyecto nuevo y de este tipo, la experiencia personal es que antes o después habrá una pérdida de motivación para mantener un ritmo de trabajo óptimo.
7. **Compaginación con los estudios académicos. (Factor Tiempo).** El proyecto se desarrolla paralelamente al último curso de los estudios universitarios, debiendo ser correctamente compaginado para que «nada pise a nada» y no produzca retrasos. La escasez de tiempo puede suponer un problema en caso de que en los primeros *sprints* de trabajo no se alcance un ritmo de desarrollo adecuado, la fecha límite es conocida desde el inicio del proyecto y se debe de tener en cuenta.
8. **Corrupción del alcance.** En caso de que los objetivos del proyecto no estén claramente definidos. Una correcta hoja de ruta permitirá a todos los involucrados a conocer la parametrización deseada. Estando muy relacionado con la motivación (no «ver el final» del proyecto nunca) y por consecuencia con el ritmo de trabajo.
9. **Coste de la infraestructura.** Se debe tener en cuenta que se va a desarrollar una aplicación web, pero por su naturaleza necesitará un servidor (distribuido o no) para su ejecución. A baja escala puede no ser un riesgo, pero se debe vigilar en caso de despliegue en las principales *cloud*.
10. **Falta de claridad.** La comunicación entre todas las partes implicadas debe de ser lo más fluida y natural posible, permitiendo minimizar los retrasos por tener que rehacer algo que se había especificado de una forma y no se había entendido correctamente (Inequívoco).

B.5. Catálogo de requisitos

En esta sección se van a definir de forma clara, completa, precisa y verificable todas las funcionalidades y restricciones del sistema.

A pesar de que el proyecto tiene dos «enfoques», la parte de UBUMLaaS y la parte de bibliotecas, los requisitos funcionales y no funcionales se van a desglosar juntos, siguiendo el orden en el que aparecen en este texto.

Requisitos funcionales

- **RF-1 Uso de algoritmos de aprendizaje automático.** La aplicación debe de ser capaz de entrenar un modelo entrenado con un algoritmo elegido por el usuario y posteriormente utilizar dicho modelo para predecir sobre un conjunto de datos.
 - **RF-1.1 Entrenar el modelo.** El usuario debe de poder entrenar un modelo nuevo en cada experimento.
 - **RF-1.1.1 Elección del algoritmo.** El usuario debe de ser capaz de elegir el algoritmo que considere oportuno de entre todos los posibles.
 - **RF-1.1.2 Parametrización del algoritmo.** El usuario debe poder parametrizar el algoritmo cómo considere oportuno para su problema.
 - **RF-1.1.3 Conjuntos de datos especiales.** El usuario en caso de realizar experimentos de Semi-Supervisado tendrá que utilizar conjuntos de datos preparados para ello.
 - **RF-1.2 Descarga del modelo.** El usuario debe de ser capaz de descargar el modelo para poder usarlo en otros sistemas.
 - **RF-1.3 Reutilización del modelo.** El usuario debe de ser capaz de crear un modelo utilizando una parametrización base de otro modelo existente en el sistema.
 - **RF-1.4 Predicción de nuevos prototipos.** El usuario debe de ser capaz de utilizar un modelo ya entrenado para predecir nuevos conjuntos de datos que posean la misma relación de atributos.
 - **RF-1.5 Estadísticas del entrenamiento.** El usuario debe de ser capaz de visualizar las estadísticas del experimento ejecutado, independientemente de si el entrenamiento ha sido mediante validación cruzada o con partición mediante porcentajes para entrenamiento y pruebas.
 - **RF-1.6 Consulta de experimentos.** El usuario debe de poder consultar aquellos experimentos que ha lanzado.
- **RF-2 Uso de algoritmos de selección de instancias.** El usuario deberá poder elegir si usar o no, para cualquier experimento independientemente de su naturaleza, los algoritmos de selección de instancias codificados.

- **RF-2.1 Parametrización del algoritmo.** El usuario debe poder parametrizar el algoritmo cómo considere oportuno para su problema.
- **RF-3 Administración de usuarios.**
 - **RF-3.1 Dar de alta nuevos usuarios.** El administrador debe poder crear un nuevo usuario con la información básica.
 - **RF-3.1.1 Activación del usuario.** El sistema debe mandar el correspondiente correo de activación al nuevo usuario.
 - **RF-3.1.2 Contraseña del usuario.** Se generará una contraseña complaciente con la política de seguridad de la plataforma. En ningún momento dicha contraseña podrá ser conocida por ningún administrador o miembro del sistema. El usuario deberá de restaurar la contraseña antes de iniciar sesión por primera vez.
 - **RF-3.2 Activación de usuarios.** El administrador debe de poder activar o desactivar a un usuario en concreto.
 - **RF-3.3 Hacer administrador a un usuario.** El administrador debe de poder hacer nuevos usuarios administradores.
 - **RF-3.4 Eliminar a un usuario.** El administrador debe de poder eliminar a un usuario cualquiera del sistema, independientemente de si este usuario es administrador o no.
 - **RF-3.5 Auto-Modificación del administrador.** El administrador no debe de poder desactivarse, quitarse de administrador o eliminarse a sí mismo.
- **RF-4 Modificación de datos del usuario.**
 - **RF-4.1 Datos básicos.** El usuario debe de poder modificar sus datos básicos, pero nunca pudiendo dejarlos «en blanco».
 - **RF-4.2 Datos adicionales.** El usuario debe de poder añadir, modificar o eliminar, una serie de datos adicionales.
 - **RF-4.3 Imagen de perfil.** El usuario debe de poder actualizar su foto de perfil, cumpliendo con una serie de requisitos de tamaño y formato.
 - **RF-4.4 Actualización de contraseña.** El usuario deberá de poder actualizar su contraseña en caso de considerarlo necesario.

- **RF-5 Administración del sistema en tiempo real.**

- **RF-5.1 Información de red.** El administrador debe de poder visualizar la configuración actual de red en la que la plataforma está desplegada.
- **RF-5.2 Información de carga.** El administrador debe de poder visualizar la carga del actual del sistema en términos de uso de procesador y memoria.
- **RF-5.3 Información adicional.** El administrador debe de poder visualizar datos adicionales como el uso de red, almacenamiento disponible, ...

- **RF-6 Estadísticas de uso.**

- **RF-6.1 Estadísticas de uso para usuarios.**
 - **RF-6.1.1 Uso últimos 7 días.** El usuario debe de poder visualizar unas estadísticas generales de su uso particular en los últimos 7 días naturales.
 - **RF-6.1.2 Uso de cada tipo de algoritmo.** El usuario debe de poder visualizar qué y cuántos algoritmos de cada tipo ha ejecutado. Además del tiempo de ejecución global de cada tipo.
 - **RF-6.1.3 Estadísticas generales.** El usuario debe de poder conocer cuántos experimentos ha ejecutado en total y cuántos conjuntos de datos tiene alojados en el sistema.
- **RF-6.2 Estadísticas de uso para administradores.**
 - **RF-6.2.1 Estadísticas generales.** El administrador debe de poder de un vistazo conocer el uso general que se le está dando al sistema. (Número de experimentos, número de usuarios, tipo de experimentos,...)
 - **RF-6.2.2 Uso últimos 7 días.** El administrador debe de poder conocer el número de experimentos que se han ejecutado cada día de los últimos 7 días naturales.
 - **RF-6.2.3 Distribución de los usuarios.** El administrador debe de poder conocer las estadísticas generales de uso y países de origen de los usuarios del sistema.

Requisitos no funcionales

- **RNF-1 Usabilidad.** La plataforma debe de ser fácil tanto de aprender a utilizar como clara a la hora de reportar los errores que se puedan cometer. La interfaz debe ser intuitiva.
- **RNF-2 Rendimiento.** La interfaz web no se puede quedar «colgada», además debe de tener unos tiempos de carga razonables.
- **RNF-3 Escalabilidad.** La plataforma debe soportar que se le añadan nuevas funcionalidades con relativa facilidad.
- **RNF-4 Disponibilidad.** La plataforma debe de ser accesible a través de Internet sin importar la geolocalización del cliente.
- **RNF-5 Fiabilidad.** La plataforma debe garantizar que los modelos calculados son precisos. Además de en caso de pérdidas de conexión, que no ocurran pérdidas de datos.
- **RNF-6 Seguridad.** La plataforma debe gestionar correctamente *tokens*, contraseñas, así como el control de administradores o no.
- **RNF-7 Mantenibilidad.** La plataforma debe cumplir los estándares de código de cada uno de los lenguajes en los que se desarrolla.
- **RNF-8 Soporte.** La plataforma debe dar soporte a ficheros CSV y ARFF como mínimo. Así como ser compatible con HTML5.
- **RNF-9 Monitorización.** La plataforma debe ser fácilmente monitizable por un administrador.
- **RNF-10 Internacionalización.** La plataforma debe de estar desarrollada en un inglés sencillo y fácil de comprender por todo tipo de usuarios no nativos.
- **RNF-11 Respuesta autónoma.** En caso de inicio o reinicio, el tiempo empleado por la plataforma hasta estar al 100 % de operatividad de nuevo debe ser inferior a los 3 minutos.

B.6. Especificación de requisitos

Dentro de esta sección se desarrolla el Diagrama de Casos de Uso, ver Figura B.1, y la explicación correspondiente de cada uno de ellos.

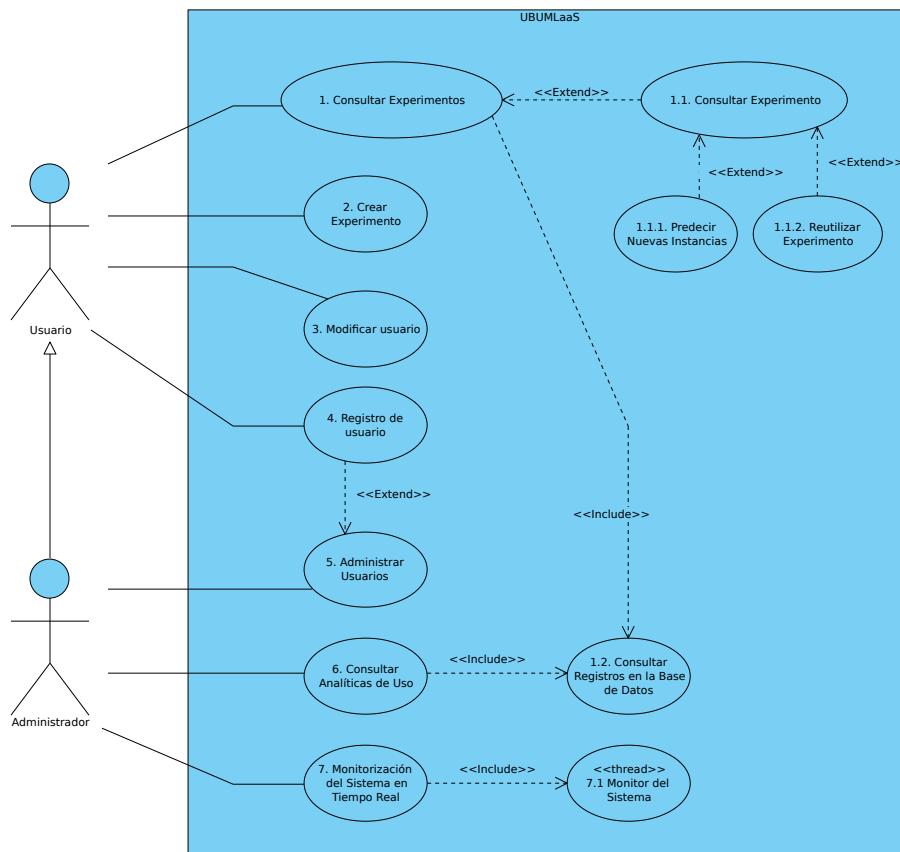


Figura B.1: Diagrama de casos de uso.

Actores

Actuarán dos actores con el sistema, un usuario (el actor general) y un administrador (actor especializado heredado del usuario).

Casos de uso

A continuación, se detallan las tablas correspondientes a los casos de uso anteriormente planteados, en orden.

CU-1	Consultar Experimentos
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1.3, RF-1.5, RF-1.6
Descripción	Permite al usuario consultar sus experimentos y reutilizarlos.
Precondición	El sistema de colas se encuentra en ejecución.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Por cada experimento lanzado se da la opción de ver detalle, reutilizar o eliminar.
Postcondición	El número de experimentos mostrados al usuario es igual al número de experimentos asociados con ese ID en la base de datos.
Excepciones	No existen excepciones posibles.
Importancia	Alta

Tabla B.1: CU-1 Consultar Experimentos.

CU-1.1	Consultar Experimento
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1.2, RF-1.3, RF-1.4, RF-1.5
Descripción	Permite al usuario consultar un experimento en concreto, si ha finalizado, junto con las métricas reportadas.
Precondiciones	<ul style="list-style-type: none"> ■ El experimento existe. ■ En caso de haber finalizado y tener métricas de rendimiento, se cargan.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Por cada experimento lanzado se da la opción de ver detalle, reutilizar o eliminar. 4. Dentro de ver en detalle puede predecir nuevas instancias o descargar el modelo, así como consultar las métricas resultantes. En caso de haber fallado se muestra el motivo del fallo.
Postcondición	El identificador del experimento no varía independientemente de lo que el usuario haga con él.
Excepciones	No existen excepciones posibles.
Importancia	Media

Tabla B.2: CU-1.1 Consultar Experimento.

CU-1.1.1 Predecir Nuevas Instancias	
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1.4
Descripción	Permite al usuario predecir nuevas instancias en función a un modelo previamente entrenado.
Precondiciones	<ul style="list-style-type: none"> ■ El experimento existe y ha finalizado. ■ Las nuevas instancias a predecir tienen los mismos atributos que con las que se entrenó el modelo.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Por cada experimento lanzado se da la opción de ver detalle, reutilizar o eliminar. 4. Dentro de ver en detalle hace <i>click</i> en «Predict». 5. Sube el conjunto de datos a predecir. 6. Se le muestra al usuario el resultado de la predicción.
Postcondición	El modelo no se ha visto afectado por el proceso de predicción.
Excepciones	El conjunto de datos pasado no cumple con los requisitos para el modelo.
Importancia	Alta

Tabla B.3: CU-1.1.1 Predecir Nuevas Instancias.

CU-1.1.2	Reutilizar Experimento
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1, RF-1.1, RF-1.1.1, RF-1.1.2, RF-1.1.3, RF-1.3
Descripción	Permite al usuario reutilizar el experimento que ya había creado.
Precondición	El experimento base existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Busca el experimento del que desea obtener la parametrización para uno nuevo. 4. Hace <i>click</i> en «Reuse».
Postcondiciones	<ol style="list-style-type: none"> 1. El modelo base no se ha visto afectado. 2. El usuario se encuentra en la pantalla «Nuevo Experimento» con la configuración «nueva».
Excepciones	<ul style="list-style-type: none"> ■ Algún parámetro interno ha cambiado y ya no se puede reutilizar el experimento. ■ Solo se puede recuperar parte de la configuración.
Importancia	Media

Tabla B.4: CU-1.1.2 Reutilizar Experimento.

CU-1.2	Consultar Registros en la Base de Datos
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6
Descripción	Recuperación de los registros necesarios de la base de datos.
Precondición	Existen los registros.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Realiza alguna acción relacionada con la base de datos. 3. Se le devuelven los datos solicitados, y/o 4. Se almacenan los nuevos datos.
Postcondición	La integridad de la base de datos no se ha visto afectada.
Excepciones	Intento de escritura simultánea por parte de dos usuarios.
Importancia	Alta

Tabla B.5: CU-1.2 Consultar Registros en la Base de Datos.

CU-2	Crear Experimento
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-1, RF-1.1, RF-1.1.1, RF-1.1.2, RF-1.1.3, RF-2, RF-2.1
Descripción	Permite al usuario crear un nuevo experimento.
Precondición	La parametrización es correcta.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Nuevo Experimento». 3. Rellena el formulario en función de un conjunto de datos y una técnica de aprendizaje automático. 4. Hace <i>click</i> en «Crear».
Postcondiciones	<ol style="list-style-type: none"> 1. El experimento ha sido añadido a las colas de ejecución. 2. El usuario recibe un correo electrónico con la finalización del experimento.
Excepciones	El experimento ha sido incorrectamente parametrizado y se ha levantado una excepción al intentar ejecutarlo.
Importancia	Alta

Tabla B.6: CU-2 Crear Experimento.

CU-3	Modificar Usuario
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.3, RF-4.4
Descripción	Permite al usuario modificar sus datos personales dentro de la plataforma.
Precondición	Los datos del usuarios son recuperados de la base de datos.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Hace <i>click</i> en «Editar Perfil». 4. Modifica los datos como considere oportuno. 5. Hace <i>click</i> en «Guardar».
Postcondiciones	<ol style="list-style-type: none"> 1. Todos los campos son validados de forma que individualmente cumplan sus respectivas restricciones de formato. 2. Para aquellos campos que deben ser únicos, se garantiza su unicidad. 3. Los datos actualizados son visibles desde el momento en el que se actualiza la página para el usuario.
Excepciones	Modificación concurrente de la base de datos.
Importancia	Baja

Tabla B.7: CU-3 Modificar Usuario.

CU-4	Registro de Usuario
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-3, RF-3.1, RF-3.1.1, RF-3.1.2
Descripción	Permite al administrador crear un nuevo usuario, o a un cliente registrarse en la plataforma y convertirse en administrador.
Precondiciones	No existen precondiciones.
Acciones	<ol style="list-style-type: none"> 1. El administrador entra en la plataforma. 2. Hace <i>click</i> en «Usuarios» en el panel lateral de administración. 3. Hace <i>click</i> en «Nuevo Usuario». 4. Introduce los datos del nuevo usuario. 5. Hace <i>click</i> en «Guardar».
Postcondiciones	<ol style="list-style-type: none"> 1. Todos los campos son validados de forma que individualmente cumplan sus respectivas restricciones de formato. 2. Para aquellos campos que deben ser únicos, se garantiza su unicidad. 3. El nuevo usuario recibe un correo electrónico con el <i>token</i> de activación de la cuenta.
Excepciones	Modificación concurrente en la base de datos.
Importancia	Baja

Tabla B.8: CU-4 Registro de Usuario.

CU-5	Administrador Usuarios
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-3, RF-3.1, RF-3.1.1, RF-3.1.2, RF-3.2, RF-3.3, RF-3.4, RF-3.5
Descripción	Permite al administrador crear, (de)activar, hacer (o quitar de) administrador, o eliminar a un usuario.
Precondición	El usuario a modificar no es el mismo usuario que está modificando.
Acciones	<ol style="list-style-type: none"> 1. El administrador entra en la plataforma. 2. Hace <i>click</i> en «Usuarios» en el panel lateral de administración. 3. Puede buscar si así lo desea al usuario en cuestión. 4. Realiza las modificaciones pertinentes.
Postcondición	La modificación ha sido correcta.
Excepciones	Modificación concurrente en la base de datos.
Importancia	Media

Tabla B.9: CU-5 Administrar Usuarios.

CU-6	Consultar Analíticas de Uso
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-6, RF-6.1, RF-6.1.1, RF-6.1.2, RF-6.1.3, RF-6.2, RF-6.2.1, RF-6.2.2, RF-6.2.3
Descripción	Permite a un usuario comprobar sus estadísticas de uso. Y si es administrador, las de la plataforma.
Precondición	Existen estadísticas que mostrar.
Acciones (para el usuario)	<ol style="list-style-type: none"> 1. El usuario entra en la plataforma. 2. Hace <i>click</i> en «Mis Experimentos». 3. Hace <i>click</i> en «Estadísticas». 4. Puede visualizar las estadísticas generales de la plataforma.
Acciones (para el administrador)	<ol style="list-style-type: none"> 1. El administrador entra en la plataforma. 2. Hace <i>click</i> en «Dashboard» en el panel lateral de administración. 3. Puede visualizar las estadísticas generales de la plataforma.
Postcondiciones	No existen postcondiciones.
Excepciones	No existen excepciones.
Importancia	Alta

Tabla B.10: CU-6 Consultar Analíticas de Uso.

CU-7	Monitorización del Sistema en Tiempo Real
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-5, RF-5.1, RF-5.2, RF-5.3
Descripción	Permite al administrador comprobar el estado de carga actual del sistema.
Precondición	Existen registros de datos para calcular las estadísticas que mostrar.
Acciones	<ol style="list-style-type: none"> 1. El administrador entra en la plataforma. 2. Hace <i>click</i> en «<i>Live Monitor</i>» en el panel lateral de administración. 3. Puede visualizar las estadísticas generales de la plataforma.
Postcondiciones	No existen postcondiciones.
Excepción	En caso de que no existan datos aún.
Importancia	Alta

Tabla B.11: CU-7 Monitorización del Sistema en Tiempo Real.

CU-7.1	Monitor del Sistema
Versión	1.0
Autor	Daniel Puente Ramírez
Requisitos asociados	RF-5.1, RF-5.2, RF-5.3
Descripción	Proceso de recolección de información de uso del sistema.
Precondición	Glances está instalado en el sistema.
Acciones	Ninguna, ejecución en paralelo en el sistema.
Postcondiciones	No existen postcondiciones.
Excepción	No existen excepciones.
Importancia	Alta

Tabla B.12: CU-7.1 Monitor del Sistema.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se va a exponer cómo se han resuelto los objetivos anteriormente comentados. Así como la definición de datos que se utilizan en la aplicación, procedimientos, etc.

C.2. UBUMLaas

Diseño de datos

La aplicación cuenta con las siguientes entidades:

- **Usuarios (Users).** Posee toda la información relacionada con los usuarios. Almacenando su identificador único en el sistema, su correo electrónico, usuario, contraseña *hasheada*, país y uso que ha indicado que va a dar a la aplicación, además de si se encuentra activo o no, o del tipo de usuario que es (administrador o usuario normal).

Como campos adicionales puede almacenar la página web del usuario, algunas redes sociales como son Twitter, LinkedIn, GitHub. Junto con la institución a la que pertenece y su Google Scholar.

- **Algoritmos (Algorithms).** Guarda la información relacionada con cada algoritmo que se puede utilizar, teniendo un identificador único, un nombre de algoritmo para uso interno, el nombre que se mostrará en la web, así como los parámetros de configuración y a qué biblioteca pertenece.

- **Filtros (Filters).** Guarda la información relacionada con cada filtro que se puede utilizar, teniendo un identificador único, un nombre de filtro para uso interno, el nombre que se mostrará en la web, así como los parámetros de configuración y a qué biblioteca pertenece.
- **Experimentos (Experiments).** Almacena toda la información de un experimento lanzado. Posee un identificador de experimento único, el identificador del usuario que lo lanzó, el nombre (interno) del algoritmo en cuestión, junto con la configuración de este, y forma homónima para los filtros (en caso de utilizar un filtro). Referencia a los datos de entrenamiento, y en caso de haber terminado, los resultados del experimento.
Posee dos *timestamps* representando la hora de inicio y fin del entrenamiento, un campo adicional representa el estado que tiene. Junto con todos estos datos se almacena la configuración del experimento.
- **Países (Countries).** Recoge toda la información que puede ser útil a la hora de trabajar con países. Posee el nombre oficial del país en cuestión, así como la representación en Alpha 2¹ y 3, el número de identificación único de cada país, además, la longitud y latitud de la capital del país.

¹Los códigos alfa-2 son códigos de dos letras definidos en la norma ISO 3166-1, utilizados para designar países territorios independientes y zonas geográficas especiales. Son utilizados principalmente en los dominios geográficos de primer nivel en Internet, además de direcciones postales.

Diagrama E/R

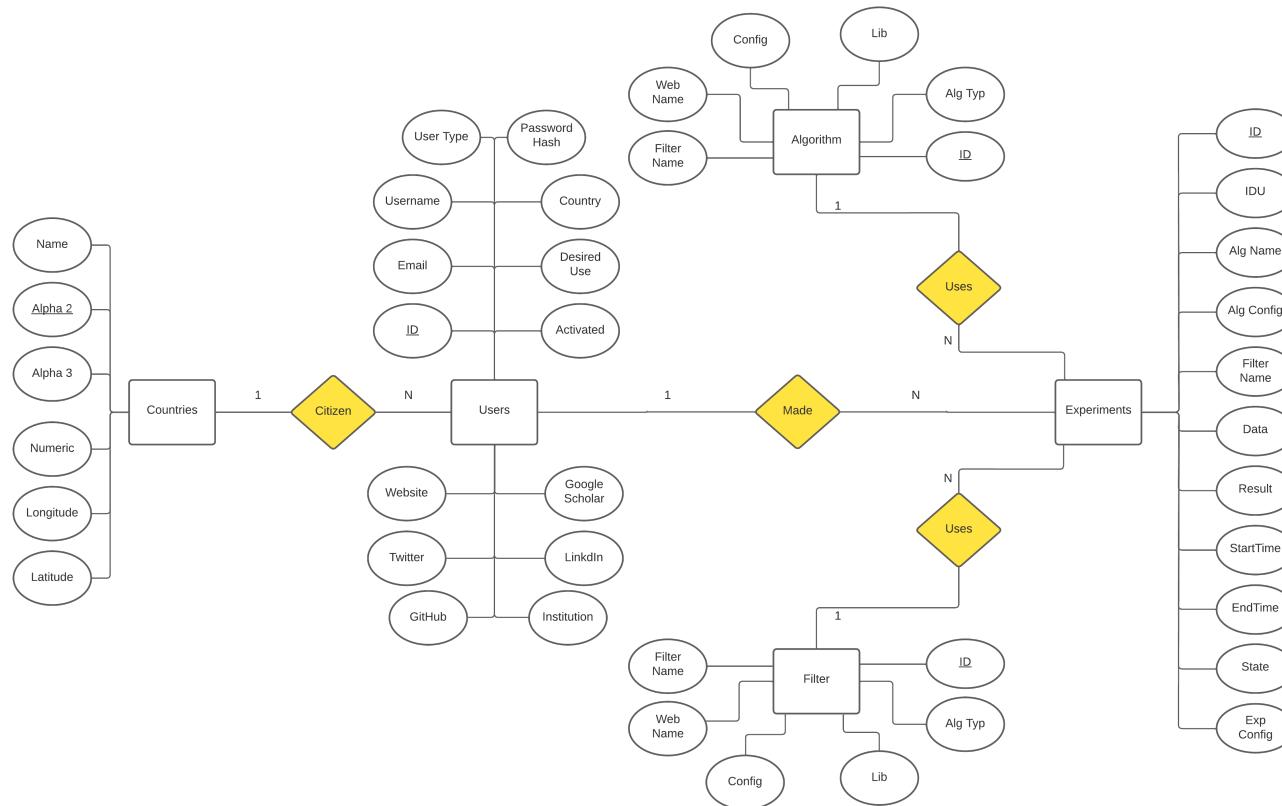


Figura C.1: Diagrama entidad relación.

Diagrama Relacional

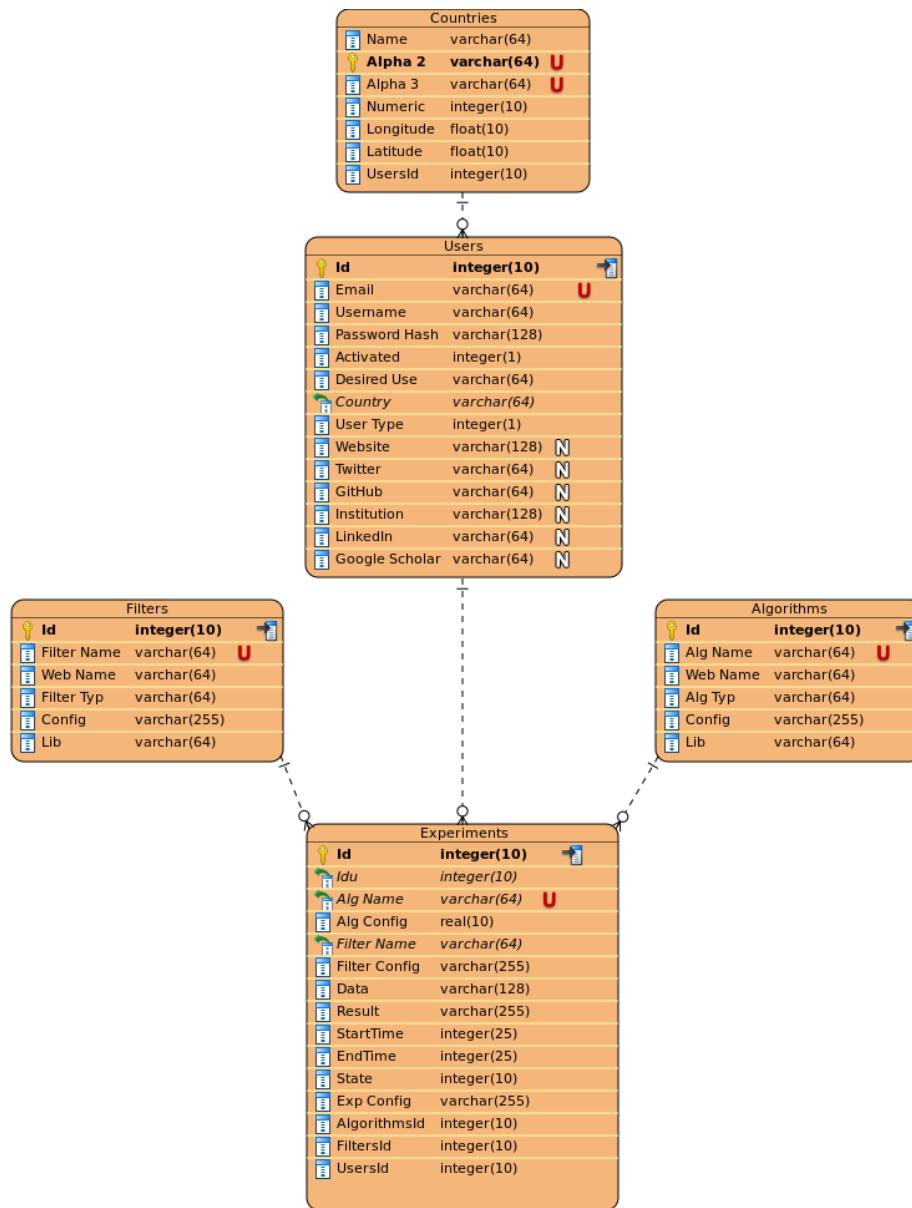


Figura C.2: Diagrama relacional.

Diseño procedural

En esta sección interna se recogen los detalles más relevantes en cuanto a los procedimientos llevados a cabo por la plataforma en función de las acciones del usuario.

A continuación se explican los diagramas de secuencia (DS):

- **DS para la monitorización del sistema en tiempo real.** Figura C.3. Muestra el proceso seguido por el sistema en el momento en el que solicita la vista correspondiente. Es el único diagrama en el que se muestra la comprobación de si es administrador o no, por brevedad en el resto se indica en forma de texto nada más.

Cuando el sistema recoge la solicitud de visualización busca los datos necesarios en la base de datos, y en el caso de no haber pasado todavía 10 minutos (valor umbral) del inicio del sistema, buscará un fichero de histórico en el que se guardan los últimos 6 meses de datos como máximo. Se procesan los datos ya que existen muchos más de los que el sistema mostrará y se devolverá la página HTML.

- **DS para las estadísticas generales (System Analytics).** Figura C.4. Necesita privilegios de administrador, omitido en el diagrama por claridad. Debido a la multitud de operaciones que debe realizar, posee una pantalla de carga que se muestra al usuario en lo que el sistema prepara la visualización.

Internamente se recorre prácticamente la base de datos en su totalidad y se obtienen las estadísticas correspondientes. En el momento en el que se tienen todos los valores calculados se guardarán en ficheros temporales que se leerán y al poco tiempo un recolector de basura los eliminará.

- **DS para crear un experimento.** Figura C.5. Cuando un usuario accede a la vista de crear un nuevo experimento, prácticamente cada botón y desplegable tienen repercusión directa en el sistema.

En la elección/subida de un conjunto de datos, se harán operaciones de lectura/escritura respectivamente sobre un directorio específico en el que se encuentran almacenados.

Para la selección de algoritmos y filtros, una vez se selecciona el tipo de algoritmo a utilizar, se leen de la base de datos aquellos algoritmos y filtros compatibles y se muestran al usuario para su elección. Una vez

seleccionados se renderizan los parámetros de configuración particulares de cada uno de ellos.

Finalmente, el usuario mandará crear el experimento, pasando al lado del servidor la ejecución de este, ver Figura C.6.

- **DS para la ejecución de un experimento.** Figura C.6. En el momento en el que el usuario manda crear el experimento, se le muestra la pantalla en la cual aparecerán los resultados, pero con un GIF indicando que aún no ha terminado la ejecución.

El servidor recoge la configuración indicada por el usuario para realizar el experimento y lo encola en las colas de ejecución `high-ubumlaas`, en las que cuando estén disponibles, realizarán el experimento según la configuración recibida.

En el momento en el que el experimento finalice, la cola pasará a ejecutar el siguiente experimento (de existir), y se le devolverá el control al sistema, este último se encargará de almacenar los resultados en la entrada correspondiente al experimento en la base de datos, de tal manera que cuando el usuario proceda a ver los resultados pueda visualizarlos. Finalmente mandará un correo electrónico al usuario «dueño» del experimento indicando que ha finalizado.

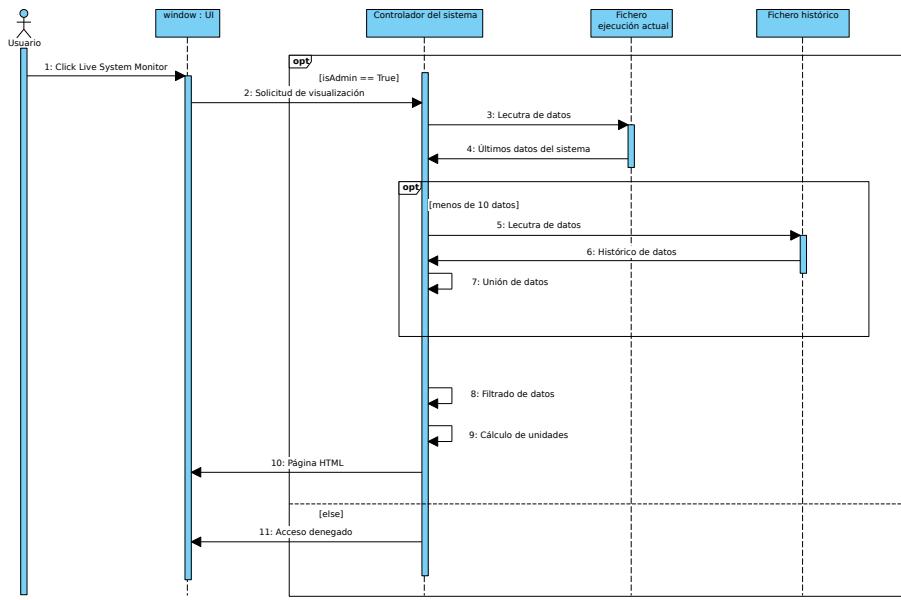


Figura C.3: Diagrama de secuencia de la monitorización en tiempo real.

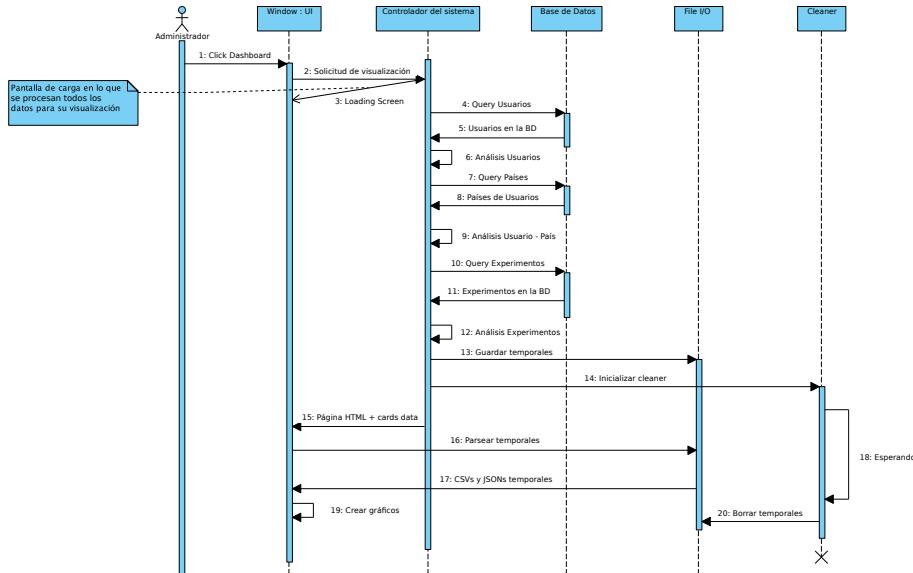


Figura C.4: Diagrama de secuencia de las estadísticas generales de la aplicación.

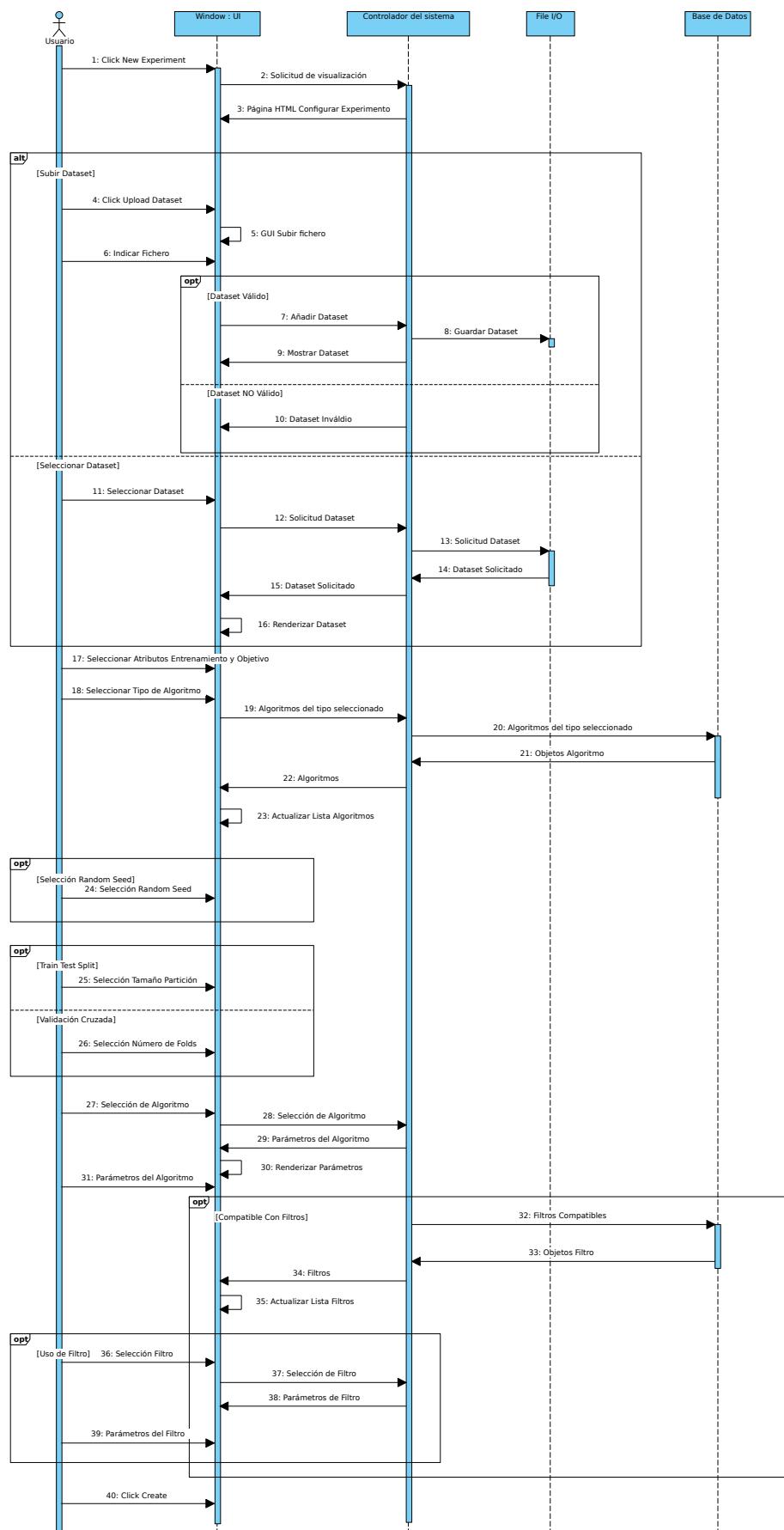


Figura C.5: Diagrama de secuencia de la creación de un nuevo experimento por parte del usuario.

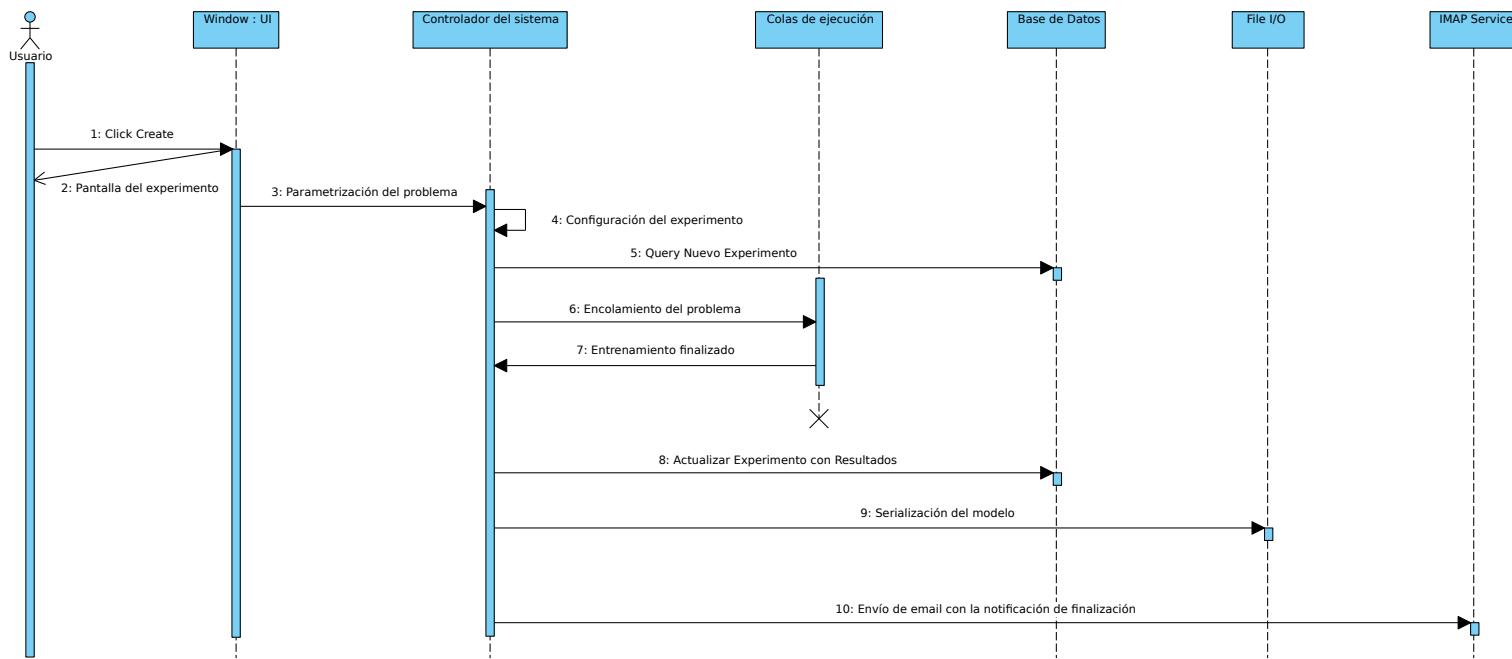


Figura C.6: Diagrama de secuencia de la ejecución de un nuevo experimento.

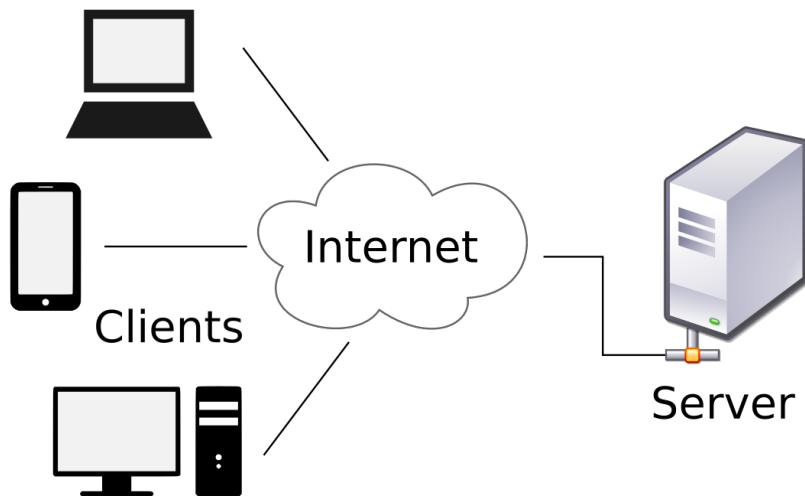


Figura C.7: Arquitectura cliente-servidor.

Diseño arquitectónico

La aplicación con el fin de cumplir con todos los requerimientos funcionales así como objetivos principales, y por ende, conseguir un bajo acoplamiento y una alta cohesión. Sigue una arquitectura de cliente servidor.

En la Figura C.7 se aprecia un modelo simplificado de la arquitectura seguida, en la cual los procesos se van a dividir en dos grupos.

- Servidor. Implementa el servicio de UBUMlaaS.
- Cliente. Solicitará los servicios de proporcionados por el servidor.

Arquitectura de tres capas

La aplicación sigue una arquitectura de tres capas (multicapa), siguiendo esta arquitectura el cliente implementa la lógica de presentación (es un cliente ligero); el servidor de aplicación implementa la lógica de negocio, y los datos residen en una base de datos de **SQLite**, por definición de la arquitectura de tres capas sería necesario que hubiera un servidor dedicado a la comunicación con la base de datos, pero ahí es donde reside una de las cualidades de **SQLite**, es *serverless*, permitiendo una auto-gestión y soportando múltiples clientes realizando tareas en paralelo.

UBUMlaaS sigue esta arquitectura por las siguientes razones:

- Desacoplamiento, cambios en la interfaz de usuario o en la lógica de la aplicación son independientes entre sí, favoreciendo la evolución de la aplicación hacia nuevos requerimientos.
- Se minimizan los cuellos de botella de la red, la información transmitida es únicamente la solicitada.
- El cliente está separado (aislado) de la base de datos, pudiendo acceder de manera sencilla a los recursos sin necesidad de conocer la ubicación de los datos.

C.3. IS-SSL

Diseño de datos

En esta sección se van a seleccionar las representaciones lógicas de los datos, las estructuras de datos utilizadas.

Los algoritmos implementados se encuentran divididos en dos bibliotecas, la separación se realiza en base al criterio lógico de qué hacen los algoritmos de cada una de ellas. Por un lado, están los algoritmos de selección de instancias, y por el otro, los algoritmos de aprendizaje semi-supervisado.

Todos los algoritmos utilizan la clase auxiliar *Nearest Neighbors* de Scikit-Learn [9] para el cálculo de los vecinos cercanos. Teniendo en cuenta que la distancia a sí misma es cero, se han codificado los algoritmos para evitar que un prototipo posea como vecino más cercano a sí mismo.

Diseño procedural

A continuación, se recogen los detalles para poder hacer uso de los algoritmos.

Todos los algoritmos se pueden utilizar de la misma manera que se esperaría al utilizar los propios de la biblioteca de Scikit-Learn. Por lo tanto, una vez están importados los correspondientes se utilizarán:

- Algoritmos de selección de instancias.
 1. Instanciar el objeto a utilizar, pasando los parámetros de configuración deseados.
 2. Pasar el conjunto de datos al método **filter** del modelo.

3. Recoger los resultados.
- Algoritmos de aprendizaje semi-supervisado.
 1. Instanciar el objeto a utilizar, pasando los parámetros de configuración, así como referencias a algoritmos de clasificación si no se quieren utilizar los proporcionados «por defecto».
 2. Pasar al método `fit` el conjunto de datos, indicando aquellas instancias para las que no se conoce la clase, con su clase a -1.
 3. Pasar al método `predict` el conjunto de datos a predecir. Obteniendo las etiquetas para el conjunto de datos pasado.

NOTA. Todas las entradas son objetos de tipo `DataFrame` de la biblioteca `Pandas`. Las salidas cuándo son vectores de una dimensión, son `arrays` de `NumPy`, si son de más de una dimensión, `DataFrames`.

Diseño arquitectónico

Debido a que se trata de una serie de bibliotecas de algoritmos, no son lo suficientemente grandes como para aplicar patrones de diseño los cuales proporcionen algún tipo de ventaja significativa.

Diseño en paquetes

Para la organización de los diferentes archivos que componen las bibliotecas se ha seguido la estrategia de *package per feature approach* (paquete por característica).

Esta estrategia permite agrupar todos los archivos en función de la funcionalidad que aportan, aumentando la legibilidad del árbol de paquetes, su modularización, así como su desarrollo continuo y ampliación de algoritmos soportados.

Ambas bibliotecas incluyen el directorio interno de `utils`. El cual proporciona clases y métodos necesarios, comunes a varios algoritmos de la biblioteca principal.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se va a describir con detalle la documentación técnica de programación. Se describirá la estructura de directorios que posee, la instalación del propio entorno de desarrollo, cómo llevar a cabo su compilación, instalación y ejecución; además de las pruebas que se han realizado.

Se debe recordar que el proyecto se encuentra dividido en dos repositorios diferenciados, UBUMLaaS e IS-SSL¹; es por ello que, se dividirá en dos secciones respectivamente, y tantas subsecciones como son necesarias para cada uno de ellos.

Ambos repositorios son consultables desde:

- **UBUMLaaS.** <https://github.com/dpr1005/UBUMLaaS>
- **IS-SSL.** <https://github.com/dpr1005/Semisupervised-learning-and-instance-selection-methods>

D.2. UBUMLaaS

Estructura de directorios

La estructura del repositorio es la siguiente:

¹Biblioteca de algoritmos de selección de instancias y aprendizaje semi-supervisado programado.

- `/`: raíz del proyecto, aquí se encuentra el README, la licencia, los ficheros de configuración de las pruebas de integración y despliegue continuo (CI-CD), junto con los ficheros de requisitos para `conda` y `pyenv`.
- `/lib/`: bibliotecas utilizadas por el sistema.
- `/lib/is_ssl`: biblioteca propia de métodos de selección de instancias y aprendizaje semi-supervisado.
- `/lib/scikit_ml_learn_data/meka/meka-release-1.9.2/`: biblioteca Meka en su versión 1.9.2.
- `/lib/skmultilearn/`: biblioteca `scikit-multilearn`.
- `/lib/unofficial_weka_packages/`: algoritmos de ADMIRABLE.
- `/lib/wekafiles/`: algoritmos concretos de `weka`.
- `/test/*`: ficheros de prueba CI-CD.
- `/ubumlaas/`: directorio principal de la plataforma.
- `/ubumlaas/admin/`: contiene toda la parte de *backend* de administración.
- `/ubumlaas/core/`: contiene el *backend* de las vistas de índice y acerca de.
- `/ubumlaas/default_datasets/`: conjuntos de datos por defecto que se añaden a los nuevos usuarios.
- `/ubumlaas/error_pages/`: contiene el *backend* de las vistas de error.
- `/ubumlaas/experiments/`: contiene el *backend* para la realización de experimentos.
- `/ubumlaas/experiments/algorithm/`: contiene las métricas para el análisis del modelo entrenado.
- `/ubumlaas/experiments/execute_algorithm/`: contiene opciones de ejecución para cada biblioteca.
- `/ubumlaas/experiments/views/`: control de las vistas relacionadas con los experimentos.
- `/ubumlaas/jobs/`: descripción de *RQ Worker Builder*.
- `/ubumlaas/static/`: contiene los ficheros estáticos de la plataforma.
- `/ubumlaas/static/avatars/`: contiene las imágenes de perfil de cada usuario.
- `/ubumlaas/static/css/`: contiene el código CSS del *frontend*.
- `/ubumlaas/static/img/`: contiene las imágenes que aparecen en la plataforma.
- `/ubumlaas/static/js/`: contiene el código JavaScript del *frontend*.
- `/ubumlaas/templates/`: ficheros HTML.
- `/ubumlaas/templates/admin/`: ficheros web de administración.
- `/ubumlaas/templates(blocks)/`: ficheros web de bloques que se añaden sobre otros documentos web.

- `/ubumlaas/templates/error_pages/`: ficheros web de errores (403, 404, ...)
- `/ubumlaas/templates/modals/`: ficheros para la representación de modales.
- `/ubumlaas/users/`: contiene el *backend* de las actividades relacionadas con el usuario.
- `/ubumlaas/weka/`: contiene los ficheros de configuración de Weka y su VM.

Manual del programador

En esta subsección se describen todos aquellos recursos utilizados por el equipo de desarrollo para, valga la redundancia, desarrollar el proyecto. De tal forma que un futuro desarrollador/mantenedor del proyecto no tenga inconvenientes a la hora de retomar el proyecto y conocerlo.

Entorno de desarrollo

Para poder continuar con el desarrollo del proyecto, se requiere tener instalado el siguiente *software* en el equipo:

- Python 3.7+.
- Bibliotecas Python.
- Git
- VSCode

En los siguientes apartados se detalla la instalación de cada uno de los componentes anteriormente citados.

Python 3.7+

Al comienzo del proyecto, muchas de sus funcionalidades eran compatibles con Python 2, pero el nuevo desarrollo ha utilizado indistintamente métodos existentes en versiones anteriores de Python y algunos que se han introducido a partir de la versión 3.7, disponible desde [4]. Es importante que los binarios se encuentren en el `path` del sistema para que no de problemas de ejecución.

Bibliotecas Python

A continuación (ver Tabla D.1), se van a detallar uno de los puntos más importantes para poder «hacer funcionar» el proyecto, puesto que se van a

necesar versiones concretas de determinadas bibliotecas para que todo se integre correctamente con todo y se pueda ver y utilizar como un sistema homogéneo.

Las versiones indicadas en la tabla D.1 son las que se han utilizado para el desarrollo del proyecto, se pueden actualizar a versiones futuras, siempre y cuando sean compatibles entre sí.

Compilación, instalación y ejecución del proyecto

En esta subsección se va a detallar el proceso a seguir para poder hacer uso del proyecto en local, modificarlo y/o utilizarlo. La forma de desarrollo del proyecto no ha sido estrictamente en local, sino que el proyecto se encontraba alojado en un equipo servidor y mediante SSH³ se realizaba la conexión y posterior edición de los ficheros.

Adquisición del código fuente

Lo primero que se necesita es obtener el código en el equipo, para ello podemos seguir una de las siguientes aproximaciones:

- Mediante el uso de la terminal.
 1. Apertura de la terminal.
 2. Desplazarse al directorio en donde se desee clonar el repositorio (usando `cd` en Unix o `dir` en Windows).
 3. Hacer uso del siguiente comando:
`git clone https://github.com/dpr1005/UBUMLaaS.git`
 4. Se dispone de una copia idéntica a la alojada en el repositorio de GitHub.
- Descarga desde el navegador.
 - Apertura del navegador preferido.
 - Introducir en la barra de búsqueda la siguiente dirección:
`https://github.com/dpr1005/UBUMLaaS/archive/refs/heads/master.zip`
 - Aceptar la descarga en caso de tener habilitada la comprobación.
 - Navegar con el Explorador de archivos del sistema hasta el directorio de descarga.

³Suite de protocolos los cuales especifican estándares para operar los servicios de red de forma segura entre anfitriones para los que no existe una relación de confianza a través de redes no seguras. Las comunicaciones entre pares se encuentran encriptadas.

Biblioteca	Versión	Descripción
<code>email-validator</code>	1.1.1	Validar direcciones de correo electrónico.
<code>flask</code>	1.1.1	Web <i>framework</i> .
<code>flask-login</code>	0.4.1	Control usuarios y sesiones en Flask.
<code>flask-mail</code>	0.9.1	Envío de <i>emails</i> con Flask.
<code>flask-migrate</code>	2.5.2	Migrar SQLAlchemy DB a Flask.
<code>flask-redis</code>	0.4.0	Soporte a Redis en Flask.
<code>flask-sqlalchemy</code>	2.4.0	Soporte a SQLAlchemy en Flask.
<code>flask-wtf</code>	0.14.2	Render, validar y CSRF formularios.
<code>future</code>	0.16.0	Soporte a Python 2 y 3.
<code>geopy</code>	2.2.0	Geo-codificación.
<code>glances</code>	3.2.4.2	Monitorización del sistema.
<code>imbalanced-learn</code>	0.5.0	ML con datos desbalanceados.
<code>itsdangerous</code>	1.1.0	Paso de datos en entornos no seguros.
<code>liac-arff</code>	2.2.1	Escritura y lectura de ficheros ARFF.
<code>numpy</code>	1.22.3	Computación de <i>arrays</i> .
<code>pandas</code>	0.25.1	Estructuras de datos.
<code>psutil</code>	5.9.0	Procesar y monitorizar sistemas.
<code>pycountry</code>	22.3.5	Datos de países.
<code>pytest</code>	5.2.1	Pruebas en Python.
<code>python-weka-wrapper3</code>	0.1.7	<i>Wrapper</i> para Weka.
<code>requests</code>	2.22.0	<i>Requests</i> para humanos.
<code>rq</code>	1.1.0	Crear y procesar trabajos «de fondo».
<code>scikit-learn</code>	0.24	Módulos de minería de datos y ML.
<code>selenium</code>	3.141.0	Auto interacción con navegador web.
<code>urllib3</code>	1.25.6	Conexiones HTTP seguras.
<code>werkzeug</code>	0.15.6	Biblioteca de aplicaciones web WSGI. ²
<code>whichcraft</code>	0.4.1	Funcionalidad <i>shutil.which</i> .

Tabla D.1: Bibliotecas utilizadas por UBUMlaaS y sus versiones.

- Uso de **GitKraken**.
 - Apertura de la aplicación.
 - Hacer *click* en *Clone a repo*.
 - En *Repository Management* → *Clone* → *Clone with URL*:
 - Indicar la ruta local en la que nos interesa que se clone el repositorio.
 - En URL introducir:
`git clone https://github.com/dpr1005/UBUMLaaS.git`
 - Hacer *click* en *Clone the repo!*.

Importar el proyecto en Visual Studio Code

Se diferencian dos aproximaciones, local o como se ha operado, conexión mediante SSH.

- Importar el proyecto en la propia máquina donde se va a desplegar y será en ella en la que se edite.
 1. Apertura de **Visual Studio Code**.
 2. Hacer *click* en Abrir.
 3. Seleccionar el directorio raíz donde lo hayamos alojado.
- Importar el proyecto en una máquina y editarlo desde otra.
 1. Seguir los pasos de la adquisición del código en la máquina en la que se va a alojar el código. En el equipo local no va a estar.
 2. Apertura de **Visual Studio Code**
 3. Navegar a las Extensiones e instalar **Remote - SSH**, disponible desde [6].
 4. Instalar un cliente SSH compatible con **OpenSSH**. Ver guía [31].
 5. Con todo instalado, se realiza la conexión a la máquina remota.
 - a) Presionar F1 y correr el comando: **Remote-SSH: Open SSH Host...**
 - b) Introducir el usuario y el host/IP en el formato:
`user@host-o-ip ó user@domain@host-o-ip`
 - c) En caso de que se solicite, introducir la contraseña, pero se recomienda configurar el uso de llaves SSH, ver guía [31].

- d) Después de la conexión usar Archivo → Abrir carpeta, para abrir el directorio donde se encuentra el proyecto en la máquina remota.
- 6. Todos los cambios que se realicen se harán sobre el código en la máquina remota, la máquina local no hará más que el efecto de editor.

Crear entorno virtual de trabajo

Para poder trabajar con este proyecto (independientemente de si es para desarrollo o producción) hacen falta una serie de bibliotecas concretas de Python, las cuales, como es lógico, deben estar en la máquina en la que se va a ejecutar; dicho con otras palabras, en la que está el código. El proyecto está preparado para crear un entorno de **Conda** propio, de forma que no interfiera con otros proyectos y sea más sencillo de mantener y actualizar.

Se recomienda que los binarios de anaconda o miniconda estén configurados en el **path** del sistema para poder utilizar el comando **conda** desde la línea de comandos.

El proceso de creación del entorno virtual con **Conda** es el siguiente:

1. Apertura de la terminal.
2. Navegar hasta la raíz del proyecto.
3. Crear el entorno con:
`conda env create -f UBUMLaas_env.yml`
4. Cuando se desee utilizar se debe activar:
`conda activate UBUMLaas`

También se puede utilizar el procedimiento habitual para importar las bibliotecas al actual **venv** de la sesión de la terminal, pero se desaconseja su uso ya que un entorno «genérico» antes o después se actualizará por otros proyectos, pudiendo generar incompatibilidades con el proyecto UBUMLaas.

Instalación en Linux

Con los anteriores pasos realizados, la importación del proyecto y la activación del entorno virtual, se deben modificar una serie de ficheros con el fin de habilitar todas las funcionalidades que ofrece el proyecto.

Se deben seguir los siguientes pasos:

1. Modificar `env_variables.sh` con los valores correctos para cada uno de los campos:

```
export SECRET_KEY=<app secret key>
export EMAIL_AC=<email>
export EMAIL_PASS=<email-password>
export EMAIL_URL=<email-url>
export FLASK_ENV=development #development or production
LIBFOLDER=/absolute/path/to/UBUMLaaS
```

2. Dentro del entorno virtual de UBUMLaaS en `Conda`, se debe ejecutar el siguiente comando para permitir la importación de las variables anteriormente declaradas al entorno virtual.

```
source env_vars_to_conda.sh
```

3. Creación de la base de datos.

```
mv data_base.sqlite ubumlaas/data.sqlite
```

* En caso de poseer una base de datos con la configuración correcta, se puede poner en `./ubumlaas/` bajo el nombre de `data.sqlite`.

4. En caso de no tener instalado y configurado `Redis-Server`, ejecutar:

```
sudo apt install redis-server
sudo service redis-server start
sudo systemctl enable redis-server
```

Uso del proyecto

Lo primero de todo para poder tener el producto trabajando, es desplegarlo, para ello es requisito haber completado todos los pasos previos de esta sección. Activar el entorno virtual de `Conda`, ejecutar el lanzador, y ya está en ejecución.

```
conda activate UBUMLaaS
./run.sh
```

Nota. Es importante asegurarnos que todos los ficheros de las bibliotecas y el lanzador del proyecto poseen permisos de ejecución necesarios (en instalaciones «por defecto» de Debian, CentOS, SUSE, no debería de ser necesario más que dar permisos de ejecución al lanzador).

El usuario administrador por defecto es `Admin@AdminUBUMLaaS.es` y su contraseña es `admin4123!UBUMLaaS`.

Integración Continua

Con el objetivo de obtener como producto final un *software* de calidad, se han desarrollado una serie de pruebas de integración continua, las cuales se comprueban y analiza su resultado en cada *commit* y/o *pull request*.

Se han utilizado principalmente tres herramientas *cloud* para medir los principios de calidad del *software*.

Codacy

Herramienta la cual proporciona soporte a análisis automático del código fuente e identifica los problemas a medida que avanza. Su versatilidad permite desarrollar *software* de manera eficiente, reduciendo el número de *bugs* que se «dejan para resolver».

A través del análisis de código estático, notifica problemas de seguridad, cobertura del código, así como la duplicación y la complejidad de cada fichero en cada *commit* y *pull request*.



Figura D.1: Codacy.

Sonar Cloud

Herramienta *open source* la cual permite hacer un análisis estático del código de un proyecto, entre sus fortalezas destaca su potente capacidad de ser parametrizada, entre las acciones que realiza por defecto encontramos la detección de malas prácticas, errores de código, así como problemas de seguridad que en el pasado se han visto relacionadas con alguna CVE⁴.

A pesar de que sea un proyecto *open source* no es gratuita, y como todas las herramientas de estas características incluye una versión *community* (gratuita) para aquellos proyectos que sean *open source*.

⁴Common Vulnerabilities and Exposures, lista de fallos *software* (y *hardware*) que en el pasado se han utilizado para ganar ventaja de alguna manera.

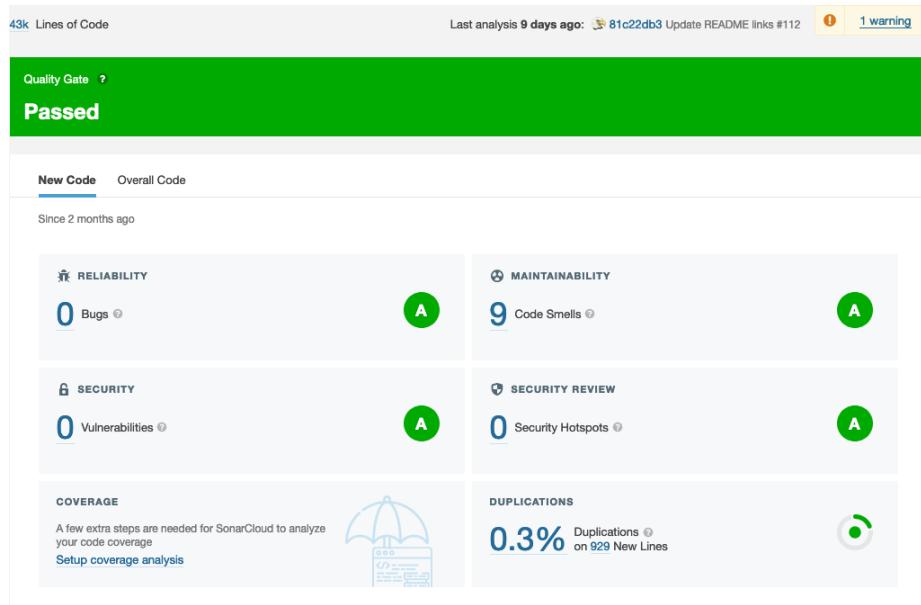


Figura D.2: SonarCloud.

Travis-CI

Herramienta *cloud* desarrollada para la realización de pruebas de integración continua sobre proyectos alojados en GitHub (con soporte beta para BitBucket, GitLab y Assembla). Permite realizar un *build* del proyecto y ejecutar sobre ella una batería de pruebas de manera automática cada vez que se realiza un *commit* y/o *pull request*, permitiendo pruebas concurrentes, incluso sobre diferentes sistemas operativos (Linux, Windows, macOS y FreeBSD).

Con el proyecto configurado en **Travis-CI** se debe configurar un fichero YAML y debe de estar en el directorio raíz, será a partir del cual se ejecuten las pruebas.

El fichero se encuentra dividido en:

- **os**: sistema/s operativo/s sobre el cuál se va/n a realizar las pruebas.
- **dist**: distribución a utilizar.
- **language**: lenguaje de programación del proyecto.
- **python**: especificación de la versión de Python que necesita.
- **node_js**: especificación de la versión de Node JS que necesita.
- **jdk**: especificación de la versión de JDK necesaria.
- **jobs**: parametrización de los trabajos que se van a ejecutar.

- **git**: profundidad del árbol de **git** que deseamos utilizar.
- **addons**: *software* «extra» que se necesiten para las pruebas.
- **services**: especificación de los servicios que se van a utilizar.
- **before_install**: definición de comandos a ejecutar antes de los incluidos en la sección **install**.
- **install**: definición de comandos de instalación de dependencias.
- **before_script**: configuración de dependencias antes de ejecutar la sección **script**.
- **script**: pruebas a realizar.

Los *logs* son públicos y consultables desde [10].

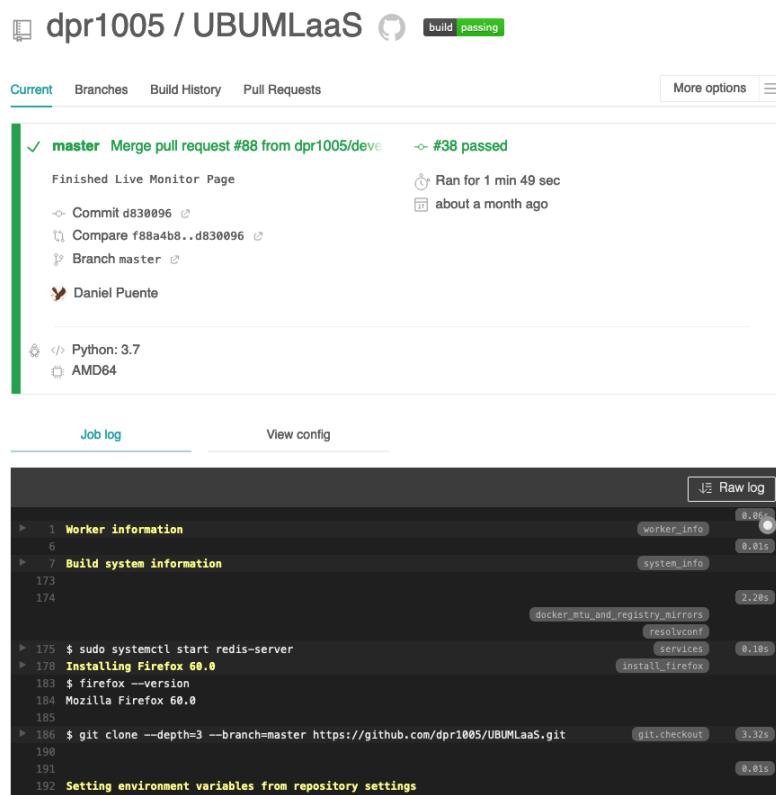


Figura D.3: Travis-CI.

Pruebas del sistema

Las pruebas del sistema se encuentran dentro de los trabajos futuros que debe seguir la aplicación.

Tal y como se ha descrito, UBUMLaaS es un *software* que ha sufrido un cambio de diseño de grandes dimensiones, por lo que las pruebas de integración continua previas que existían han dejado de ser funcionales. La problemática que surge con las existentes es que no eran pruebas sobre el *backend* de la aplicación, sino la interacción del usuario (mediante el uso de *Selenium*) con el *frontend*.

Es por ello que, al haberse hecho «de cero» toda la interfaz de la aplicación web, literalmente, no existe ninguna referencia de las antiguas que las pruebas existentes puedan reutilizar.

Debido a esta problemática, y con el fin de obtener un sistema estable, se ha seleccionado a un grupo de personas de edades comprendidas entre 18 y 55 años, profesionales de mecánica y electrónica, medicina, informática y estudiantes del grado de otras universidades españolas; con el fin de que dado un [documento](#) relativamente sencillo, puedan hacer uso de todo el sistema, y sean estos los que reporten en un [formulario de Google](#) sus impresiones y resultados. Está preparado para que se vaya leyendo el documento, probando la aplicación y posteriormente llenando el formulario, de forma que todo sea un proceso auto-guiado.

Gracias a estas respuestas, se han podido encontrar *bugs* existentes en la aplicación que no permitían trabajar correctamente con ella. El más importante de todo y el cual posee su propia [issue](#), es el desfase existente entre los algoritmos de **Scikit-Learn** de la versión 0.21 (la utilizada previamente) y la actual, la 0.24. Numerosos algoritmos han sido modificados en su parametrización y por ende en UBUMLaaS no se podían utilizar.

Durante algunos meses de desarrollo del proyecto **Travis-CI** funcionó correctamente, pero la propia herramienta sufrió una actualización a finales de marzo y dejó de ser funcional, resultando en numerosos *commits* marcados como erróneos, pero tal y como se aprecia en los propios *logs* de Travis, nunca se llegó a testar el código, sino que en la configuración de la máquina remota «rompía».

D.3. IS-SSL

Estructura de directorios

La estructura del repositorio es la siguiente:

- `/`: raíz del proyecto, aquí se encuentra el README, la licencia, los ficheros de configuración de PIP, los ficheros de configuración de las pruebas de integración y despliegue continuo (CI-CD); y, el fichero de requisitos.
- `experimentation/*`: ficheros con los que se ha realizado la experimentación y generación de resultados.
- `/datasets/*`: conjuntos de datasets en formatos `csv` y `arff`, normalizados y no normalizados.
- `/docs/`: documentación del proyecto.
- `/docs/img/`: imágenes utilizadas en la documentación.
- `/docs/img/anexos/*`: imágenes utilizadas en los anexos.
- `/docs/img/draws/`: diagramas en su formato original.
- `/docs/img/memoria/*`: imágenes utilizadas en la memoria.
- `/hypothesis/*`: primera aproximación a la investigación realizada.
- `/implementation_tests/`: conjunto de pruebas de validación sobre los algoritmos implementados.
- `/instance_selection/`: algoritmos implementados de selección de instancias.
- `/instance_selection/utils/`: métodos de apoyo comunes a los algoritmos de selección de instancias.
- `/misc/`: contiene archivos varios de formato para el repositorio (cabeceras, logos, etc.).
- `/semisupervised/`: algoritmos implementados de aprendizaje semi-supervisado.
- `/semisupervised/utils/`: métodos de apoyo comunes a los algoritmos de aprendizaje semi-supervisado.

- `/utils/`: diferentes clases y métodos de apoyo comunes tanto a selección de instancias como a semi-supervisado.

Manual del programador

En esta subsección se describen todos aquellos métodos seguidos por el equipo de desarrollo para, valga la redundancia, desarrollar el proyecto. De tal forma que un futuro desarrollador no tenga inconvenientes a la hora de retomar el proyecto.

Entorno de desarrollo

Para poder continuar con el desarrollo del proyecto, se requiere tener instalado el siguiente *software* en el equipo:

- Python 3.7+
- Bibliotecas Python
- Git
- VSCode/PyCharm/...

En los siguientes apartados se detalla la instalación de cada uno de los componentes anteriormente citados.

Python 3.7+

El desarrollo se ha realizado siguiendo las últimas formas de programación disponibles a partir de la versión 3.7 de Python. El desarrollo se comenzó después de que se dejara de mantener Python 2, por lo que se trabajó desde el inicio con versiones de Python 3. Se puede obtener la última versión disponible de Python desde [4]. Es importante que el desarrollador se asegure que los binarios han sido añadidos al `path` del sistema que esté utilizando.

Bibliotecas Python

Esta sección es la más importante de todas junto con la anterior, debido a que el proyecto depende de (está construido utilizando) bibliotecas de 3^{os}. Y en especial, determinadas versiones de las mismas.

En la Tabla D.2 se detallan las bibliotecas necesarias para utilizar el proyecto tal y como se encuentra en el repositorio. Para el uso en exclusiva de las bibliotecas de IS-SSL se deben utilizar aquellas que se encuentran en negrita.

Biblioteca	Versión	Descripción
matplotlib	3.4.3	Representación gráfica.
numpy	1.20.3	Computación de <i>arrays</i> .
pandas	1.3.4	Estructuras de datos.
scikit-learn	0.24.2	Módulos de minería de datos y ML.
scipy	1.7.1	Módulos científicos.
yagmail	0.15.277	Cliente de GMAIL.

Tabla D.2: Bibliotecas utilizadas por IS-SSL y sus versiones. Se muestran en negrita únicamente las bibliotecas necesarias para hacer uso de las bibliotecas tal y como están disponibles en PIP, las demás son utilizadas en procesos de experimentación y visualización de resultados.

Se recomienda el uso de un entorno de desarrollo de **Conda**, se facilitan ficheros de configuración tanto para **Conda** como para instalación con PIP.

Git

Para poder utilizar el repositorio ha de utilizarse el gestor de versiones **Git**. Se recomienda utilizar GUI con soporte a VC tales que no requieran de una interfaz de comandos para su utilización, pero eso se deja a decisión del futuro desarrollador.

VSCode/PyCharm/...

El desarrollo propio del producto puede ser realizado en cualquier editor de textos, incluso en **Vi** si así se desea. La ventaja de herramientas como **Visual Studio Code** o **PyCharm**, es que permiten el uso de *plugins* añadidos a los complementos del propio IDE, lo cual permite la generación de código un proceso mucho más sencillo y directo, reduciendo el número de errores ocasionados y permitiendo una depuración o refactorización del código fuente mucho más eficiente y sencilla.

Se puede obtener cada una de las herramientas desde [11, 3], respectivamente.

Compilación, instalación y ejecución del proyecto

En esta subsección se va a detallar el proceso a seguir para poder hacer uso del proyecto en local, modificarlo y/o utilizarlo.

Adquisición del código fuente

Lo primero que se necesita es obtener el código en el equipo, para ello podemos seguir una de las siguientes aproximaciones:

- Mediante el uso de la terminal.
 1. Apertura de la terminal.
 2. Desplazarse al directorio en donde se desee clonar el repositorio (usando `cd` en Unix o `dir` en Windows).
 3. Hacer uso del siguiente comando:

```
git clone https://github.com/dpr1005/  
Semisupervised-learning-and-instance-selection-  
methods.git
```
 4. Se dispone de una copia idéntica a la alojada en el repositorio de GitHub.
- Descarga desde el navegador.
 - Apertura del navegador preferido.
 - Introducir en la barra de búsqueda la siguiente dirección:
`https://github.com/dpr1005/
Semisupervised-learning-and-instance-selection-methods/
archive/refs/heads/main.zip`
 - Aceptar la descarga en caso de tener habilitada la comprobación.
 - Navegar con el Explorador de archivos del sistema hasta el directorio de descarga.
- Uso de **GitKraken**.
 - Apertura de la aplicación.
 - Hacer *click* en *Clone a repo*.
 - En *Repository Management* → *Clone* → *Clone with URL*:
 - Indicar la ruta local en la que nos interesa que se clone el repositorio.
 - En URL introducir:
`git clone https://github.com/dpr1005/
Semisupervised-learning-and-instance-selection-
methods.git`
 - Hacer *click* en *Clone the repo!*.

Importar el proyecto en PyCharm

Importar un proyecto en PyCharm es tan sencillo como:

1. Apertura de PyCharm.
2. Hacer *click* en *Open*. (Notar que también podríamos clonar el proyecto en este momento haciendo *click* en *Get from VCS*).
3. Seleccionar la ruta en el equipo dónde se encuentra el directorio raíz del proyecto.

Crear entorno virtual de trabajo

Como se ha comentado previamente, para poder trabajar con este proyecto se requieren de una serie de bibliotecas de Python. El proyecto está preparado para crear un entorno de Conda propio, de forma que no interfiera con otros proyectos y sea más sencillo de mantener y actualizar.

Se recomienda que los binarios de anaconda o miniconda estén configurados en el path del sistema para poder utilizar el comando `conda` desde la línea de comandos.

El proceso de creación del entorno virtual con Conda es el siguiente:

1. Apertura de la terminal.
2. Navegar hasta la raíz del proyecto.
3. Crear el entorno con:
`conda env create -f is-ssl.yml`
4. Cuando se desee utilizar se debe activar:
`conda activate is-ssl`

En caso de que se desee añadir al entorno (`venv`) actual en el que se encuentre el usuario:

1. Apertura de la terminal.
2. Navegar hasta la raíz del proyecto.
3. Instalar los requerimientos del proyecto con:
`pip install -r requirements.txt`

Uso del proyecto

La forma de usar la biblioteca es muy sencilla, todo IS-SSL ha sido codificado siguiendo la misma guía de estilo (PEP 8), de forma que cualquier

programador habituado con el uso de bibliotecas en Python lo encuentre intuitivo y sencillo.

Todos los métodos de selección de instancias y algoritmos de aprendizaje semi-supervisado son clases de Python, de manera que para utilizarlo hay que hacer una importación del paquete y de la clase.

Un ejemplo del uso completo de este *software* es lo encontramos en Listing E.3, donde se detallan los tipos de datos de entrada.

Según la codificación realizada, todos los métodos accesibles de las clases esperan la entrada de objetos de tipo `DataFrame` de la biblioteca de `Pandas`. Internamente en función de las operaciones que tenga que realizar, serán convertidos estos objetos a listas de Python o arreglos de `NumPy`. Independientemente de las operaciones internas, siempre la salida producida (en caso de tenerla) serán objetos de `Pandas`, no teniendo que ser necesariamente el mismo objeto de entrada modificado, en la mayor parte de las ocasiones serán objetos nuevos.

Integración Continua

Con el objetivo de diseñar un *software* lo más robusto posible, la biblioteca cuenta con una serie de pruebas de integración continua, permitiendo que todos los cambios que se vayan realizando mantengan la biblioteca en un estado correcto, indicando tempranamente aquellos posibles problemas que puedan surgir.

Este proceso se ha dividido en dos etapas, una más «adelantada» y otra más «tardía», la principal diferencia es el objetivo perseguido con cada una de las pruebas.

- Inicial. Se configuró la segunda semana de febrero, con el objetivo de comenzar a tener una visión más amplia y razonada de la calidad del código, debido a que todavía no estaban desarrollados al 100 % todos los algoritmos y aún eran susceptibles de sufrir modificaciones en tanto en cuanto a sus entradas/salidas/optimización.
- Posterior. Se configuró en Semana Santa, (segunda semana de abril), en esta fase se implementaron una batería de pruebas la cual cubre prácticamente la totalidad de IS-SSL, permitiendo obtener una base para futuras modificaciones que puedan ser necesarias realizar, asegurando que todo sigue funcionando como debería.

A pesar de que se han utilizado diferentes herramientas que se podría decir que cubren los mismos tópicos, cada una implementa las diferentes métricas de análisis de forma diferente, y por lo tanto se puede ganar en este proceso. El código cubierto no se ha configurado en cada herramienta por simplicidad, ya que ahí no hay «medias tintas» es un informe el que se genera y las herramientas lo único que proporcionan es un visor de ese fichero.

Los recursos utilizados para la integración continua se detallan a continuación.

Codacy

Herramienta la cual proporciona soporte a análisis automático del código fuente e identifica los problemas a medida que avanza. Su versatilidad permite desarrollar *software* de manera eficiente, reduciendo el número de *bugs* que se «dejan para resolver».

A través del análisis estático de código estático, notifica problemas de seguridad, cobertura del código, así como la duplicación y la complejidad de cada fichero en cada *commit* y *pull request*.

La integración es muy sencilla, basta con crearse una cuenta asociada con la de GitHub, y una vez que se ha verificado se añade la organización a la que se pertenece y el repositorio (público) que se desea comenzar a analizar. En caso de que el repositorio sea privado, pasados los 14 días de prueba se deberá actualizar la licencia de uso a una superior.



Figura D.4: Codacy.

En la Figura D.4 se aprecia uno de los gráficos que muestra Codacy, en este caso referido al número de *issues* detectados en el código analizado. Tal y como se muestra, se ha realizado un trabajo a lo largo del mes de febrero y marzo para obtener una biblioteca sin fallos aumentando su mantenibilidad.

Sonar Cloud

Herramienta *open source* la cual permite hacer un análisis estático del código de un proyecto, entre sus fortalezas destaca su potente capacidad de ser parametrizada, entre las acciones que realiza por defecto encontramos la detección de malas prácticas, errores de código, así como problemas de seguridad que en el pasado se han visto relacionadas con alguna CVE⁵.

A pesar de que sea un proyecto *open source* no es gratuita, y como todas las herramientas de estas características incluye una versión *community* (gratuita) para aquellos proyectos que sean *open source*.

El proceso de integración es sencillo, una vez registrados y asociada la cuenta con una de GitHub, se selecciona sobre qué proyecto se desea comenzar a analizar el código. El siguiente paso es definir un conjunto de reglas (si no se quiere utilizar el que se proporciona por defecto), y en cada *pull request* que se realice al repositorio, Sonar Cloud analizará todos los cambios y emitirá un informe consultable desde la web así como un comentario en la propia *pull request* con los resultados encontrados.

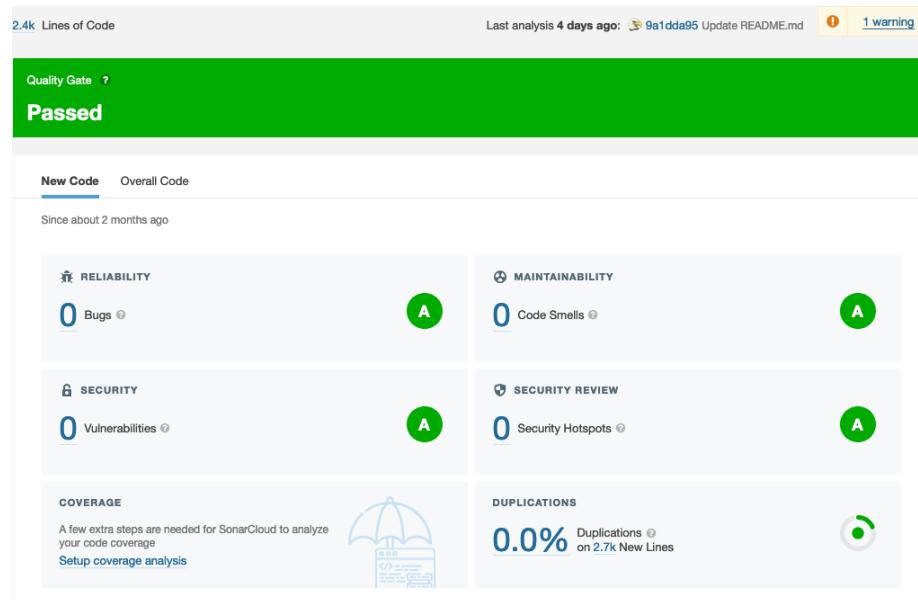
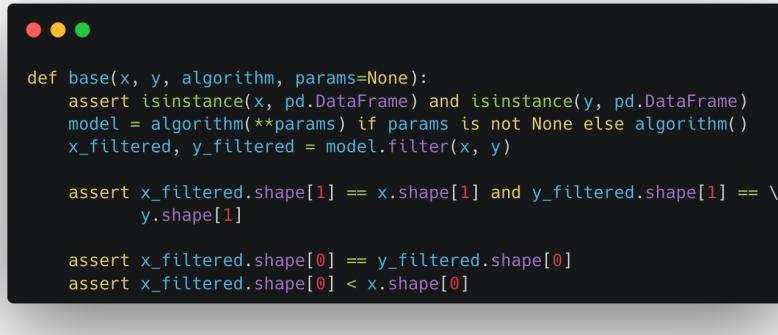


Figura D.5: SonarCloud.

⁵Common Vulnerabilities and Exposures, lista de fallos *software* (y *hardware*) que en el pasado se han utilizado para ganar ventaja de alguna manera.



```

def base(x, y, algorithm, params=None):
    assert isinstance(x, pd.DataFrame) and isinstance(y, pd.DataFrame)
    model = algorithm(**params) if params is not None else algorithm()
    x_filtered, y_filtered = model.filter(x, y)

    assert x_filtered.shape[1] == x.shape[1] and y_filtered.shape[1] == \
        y.shape[1]

    assert x_filtered.shape[0] == y_filtered.shape[0]
    assert x_filtered.shape[0] < x.shape[0]

```

Figura D.6: Prueba base para algoritmos de selección de instancias.

Pruebas del sistema

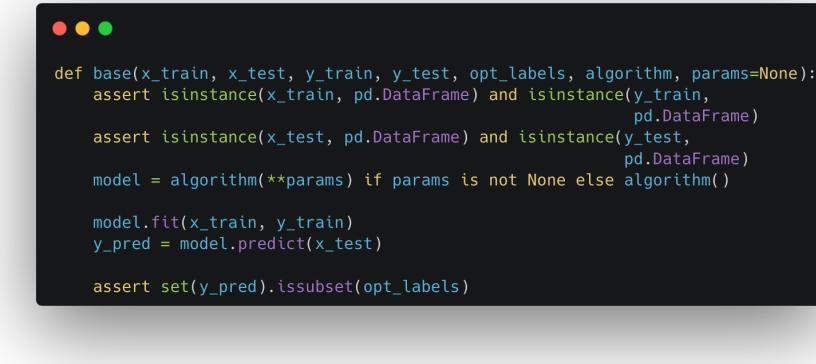
En esta sección se van a describir las pruebas que poseen las bibliotecas.

Las pruebas son realmente sencillas, ya que los propios algoritmos no poseen muchas disyunciones. Es por ello que el conjunto de pruebas codificado está compuesto de pruebas unitarias para cada algoritmo programado, permitiendo comprobar tanto que las entradas son correctas, como sus salidas. En el caso de los algoritmos de selección de instancias se asegura que las instancias recuperadas pertenecen al conjunto de datos de entrada, y que no son más que las que se introdujeron en primera instancia. De tal manera que en caso de que los algoritmos se modifiquen, el filtrado lo sigan realizando. La prueba base se puede visualizar en la Figura D.6.

Para la biblioteca de algoritmos de aprendizaje semi-supervisado se poseen también pruebas unitarias para todos y cada uno de ellos, la prueba base se puede visualizar en D.7. En ellas se comprueban:

- Tipo de objeto de entrada.
- Instanciación con mediante diferentes parámetros, tanto por defecto como mediante uso de diccionarios.
- Entrenamiento del propio algoritmo (`fit`).
- Predicción usando el modelo, `predict`.
- Las etiquetas devueltas como resultado de la predicción existen en el conjunto de entrenamiento, no se ha inventado ninguna.

Además, se ha codificado una prueba para asegurar que se siguen leyendo los ficheros ARFF correctamente.



```

def base(x_train, x_test, y_train, y_test, opt_labels, algorithm, params=None):
    assert isinstance(x_train, pd.DataFrame) and isinstance(y_train,
                                                            pd.DataFrame)
    assert isinstance(x_test, pd.DataFrame) and isinstance(y_test,
                                                            pd.DataFrame)
    model = algorithm(**params) if params is not None else algorithm()
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    assert set(y_pred).issubset(opt_labels)

```

Figura D.7: Prueba base para algoritmos de aprendizaje semi-supervisado.

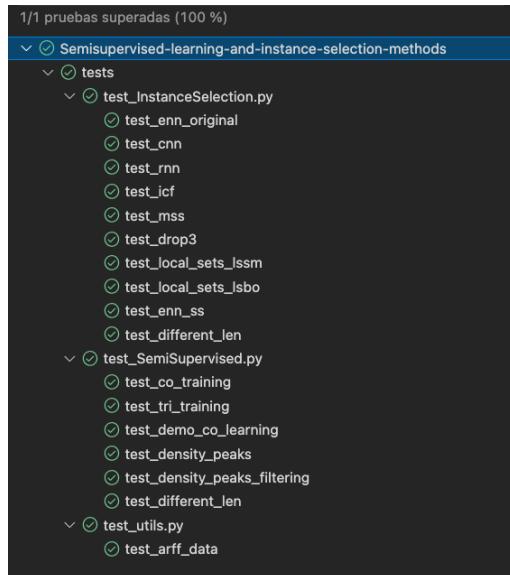


Figura D.8: Prueba base para algoritmos de aprendizaje semi-supervisado.

El resultado de las ejecuciones se puede ver la Figura D.8.

Validación

Todos los algoritmos que se incluyen en la biblioteca han sido probados y testados en múltiples iteraciones con el fin de garantizar su integridad, de forma que sea capaces de reportar los resultados adecuados y su implementación sea correcta según se propusieron en sus correspondientes *papers* de publicación.

Los algoritmos de selección de instancias han sido testados contra los homónimos propuestos por **Weka**, y en ambos se han utilizado árboles de decisión (J48 en **Weka**) y vecinos más cercanos, como clasificadores base para realizar la comparativa. La experimentación se ha realizado mediante validación cruzada, con 5 *folds*, tanto para **Weka** como para los implementados. Se ha tenido en cuenta la diferencia de lenguajes de programación Java (**Weka**) y los implementados en Python.

En las Tablas D.3 y D.4 se aprecia la comparativa uno a uno de los resultados arrojados por cada uno de los algoritmos para cada uno de los clasificadores base utilizados. Si bien puede observarse como para determinados pares conjunto de datos : algoritmo, no son compatibles por diferentes motivos, la muestra es lo suficientemente grande como para asegurar una variación menor al $\pm 5\%$ su fiabilidad.

De igual manera, los algoritmos de aprendizaje semi-supervisado han sido validados contra los propios del grupo de investigación ADMIRABLE de la Universidad de Burgos. Estos últimos sí que se encuentran implementados en Python, por lo que se esperan resultados prácticamente idénticos. La experimentación en esta ocasión también se ha realizado con validación cruzada de 5 *folds*, pero como se trata de conjuntos de datos de semi-supervisado, ha sido una validación cruzada estratificada.

En la Tabla D.5 se aprecia la comparativa uno a uno de los resultados arrojados por cada uno de los algoritmos. En el caso del *Co-Training*, la implementación desarrollada por este proyecto es capaz de soportar conjuntos de datos con más de dos vistas significativas (internamente las re-trabaja), mientras que el propuesto por ADMIRABLE no. Los resultados son tal y como se esperaba, con variaciones del $\pm 1\%$.

Dataset \ Algoritmo	CNN		RNN		ICF		MSS		DROP3	
	IS-SSL	Weka								
Contraceptive	46,71	45,48	46,71	41,34	43,86	45,14	47,32	46,83	46,13	46,36
Yeast	51,41	49,60	50,20	42,86	52,18	54,38	52,69	49,80	57,12	56,53
Wine Quality Red	51,91	51,22	48,34	53,04	52,59	53,41	54,91	51,10	57,32	56,16
Segment	87,53	83,42	78,48	75,80	90,43	90,39	93,98	92,29	87,45	92,64
Wine Quality White	47,66	48,65	42,18	45,67	53,15	49,47	51,29	50,61	51,79	51,59
Banana	87,34	87,28	86,83	77,49	81,53	87,08	88,26	86,57	85,13	89,45
Phoneme	84,92	83,98	81,46	80,03	82,60	84,09	87,27	85,16	82,70	86,09
Page Blocks	94,77	95,36	94,77	93,09	95,87	92,84	95,63	95,74	93,47	95,49
Texture	87,22	90,64	81,11	84,91	92,47	94,85	97,98	97,20	92,31	97,11

Tabla D.3: Comparación de resultados ACC de los algoritmos de selección de instancias, el clasificador base es 3-NN.

Dataset \ Algoritmo	CNN		RNN		ICF		MSS		DROP3	
	IS-SSL	Weka								
Contraceptive	51,87	51,19	51,87	41,68	45,01	50,03	51,45	51,93	49,42	50,37
Yeast	51,55	51,68	50,20	42,31	38,68	53,57	54,51	51,08	52,80	55,99
Wine Quality Red	54,98	54,41	55,41	47,84	46,72	53,35	56,22	54,91	56,35	56,41
Titanic	57,06	78,87	57,06	61,08	37,51	32,30	32,30	32,30	59,93	—
Segment	88,18	84,59	82,86	72,03	82,51	86,28	92,86	78,18	82,25	75,11
Wine Quality White	52,42	52,12	47,12	45,28	53,48	48,88	51,56	50,90	50,51	50,31
Banana	84,62	55,17	84,17	56,23	58,68	55,17	86,57	55,17	83,26	55,45
Phoneme	78,41	73,47	72,17	73,48	68,74	74,85	82,07	69,41	75,13	76,70
Page Blocks	95,08	95,91	95,08	93,24	83,95	95,60	95,63	96,02	92,69	93,95
Texture	77,02	74,87	69,75	72,24	65,53	74,62	85,27	80,16	78,49	79,05

Tabla D.4: Comparación de resultados ACC de los algoritmos de selección de instancias, el clasificador base es *Decision Tree*.

Dataset \ Algoritmo	Co-Training		Tri-Training		Democratic Co-Learning	
	IS-SSL	ADMIRABLE	IS-SSL	ADMIRABLE	IS-SSL	ADMIRABLE
Contraceptive	47,52	—	45,82	46,64	47,11	47,39
Yeast	23,47	—	13,20	23,31	51,75	52,35
Wine Quality Red	52,66	—	51,66	55,16	61,10	62,23
Titanic	77,33	77,33	77,33	77,33	78,46	78,46
Segment	66,02	—	78,01	79,91	95,58	95,93
Wine Quality White	46,18	—	42,51	44,36	59,35	62,07
Banana	60,06	61,04	62,06	61,08	87,92	88,47
Phoneme	77,50	76,00	74,24	75,98	87,12	86,94
Page Blocks	88,52	—	81,94	89,55	95,52	95,38
Texture	74,62	—	75,76	77,44	97,20	96,93

Tabla D.5: Comparación de resultados ACC de los algoritmos de aprendizaje semi-supervisado.

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección se detallan los requerimientos de la aplicación, su instalación y despliegue (en el caso de UBUMLaas) y se acompañan de una serie de indicaciones y consejos para su correcto uso.

De igual manera que en el Manual del Programador cada parte del proyecto, IS-SSL y UBUMLaas, se describirá por su propio lado, de tal manera que aunque haya aspectos comunes, cada una su propia documentación de usuario.

E.2. UBUMLaas

En esta sección se va a presentar el manual de usuario para el correcto uso de la plataforma UBUMLaas, de tal forma que sirva como guía para comprender y consiguientemente poder utilizar eficientemente el *software* desarrollado.

Requisitos de usuarios

Los requisitos mínimos para poder hacer uso de UBUMLaas son:

- Disponer de una conexión a Internet.
- Hacer uso de navegador web con soporte a HTML5.
- Tener habilitado JavaScript en el navegador.

- Tener una cuenta en la plataforma.

Instalación

Al tratarse de un producto web no se requiere de ningún tipo de instalación. Los navegadores Google Chrome, Mozilla Firefox, Safari y Microsoft Edge son soportados¹, siempre y cuando se encuentren en versiones compatibles con HTML5 y tengan activado el uso de JavaScript.

Independientemente del dispositivo de uso (ordenador de sobremesa, portátil, tableta o móvil), se requiere de conexión a Internet como es lógico. Pero no necesita de permisos adicionales, ha sido desarrollada de tal manera que utiliza la sesión local del navegador sin necesidad del uso de *cookies*.

Aunque se hizo un intento de traducción a los lenguajes más comunes, finalmente se encuentra en inglés de forma única.

Manual del usuario

En esta sección se describe como un usuario nuevo puede registrarse en la aplicación desarrollada, iniciar sesión, crear sus propios experimentos, así como predecir nuevas etiquetas con modelos ya entrenados.

Registro

Lo primero de todo una vez se conozca la URL en la que se encuentra desplegada la aplicación, es acceder a la misma, y se llegará a una página web similar a la Figura E.1, pudiendo variar en función de la resolución del dispositivo (dispositivos móviles no son recomendados pero si tabletas).

Seguidamente se procederá a hacer *click* en cualquiera de los dos botones *Register* disponibles, en el centro de la pantalla o arriba a la derecha. Ambos redireccionarán al usuario a la página de registro, la cuál será igual a la Figura E.2.

Una vez el futuro usuario de la aplicación se encuentra en frente al formulario de registro deberá de cumplimentarlo teniendo en cuenta las siguientes restricciones:

¹Todos ellos han sido probados por el equipo de desarrollo y usuarios encuestados a los que se les proporcionó un documento de uso básico y lo usaron en sus dispositivos cotidianos.



Figura E.1: Página de inicio.

A screenshot of the UBUMLaas registration page. At the top, there is a navigation bar with links for 'Home', 'About Us', 'Log In', and 'Register'. The main title is 'Register' with a sub-instruction 'You are a few keystrokes from paradise.' Below the title is a form with fields for 'Email', 'Username', 'Password', 'Confirm Password', 'Country' (set to 'Mexico'), 'Desired use' (set to 'Research'), and a 'Register' button.

Figura E.2: Página de registro.

- La dirección de correo electrónico será única en el sistema, además deberá de ser real y accesible pues a la que se mandará el correo electrónico de verificación de la cuenta.
- El usuario deberá ser único en el sistema, en caso de ya existir, se le notificará cuando se registre, use su creatividad.
- La contraseña deberá tener una longitud mínima de 8 caracteres, debiendo incluir al menos una letra mayúscula, una letra minúscula, un número y un carácter especial.
- La confirmación de la contraseña implica que se debe repetir la contraseña ingresada anteriormente.
- El país deberá ser seleccionado de la lista de países disponibles, encontrándose todos ellos en inglés, la búsqueda por teclado es soportada.
- Se deberá indicar el uso que se le va a dar a la plataforma.

Finalmente una vez se pulse el botón **Register** el cliente recibirá un correo electrónico (revisar la carpeta de correo no deseado o SPAM) con el enlace de verificación de la cuenta que acaba de crear.

Es importante tener en cuenta que mientras la cuenta no se encuentre verificada, permanecerá inactiva. En caso de tener algún problema con la activación, se recomienda contactar con soporte desde el mismo correo electrónico con el que se registró para solucionar los problemas que hayan podido surgir.

Iniciar sesión

El proceso de inicio de sesión es tan directo como hacer *click* en cualquiera de los botones dedicados a ello en el índice principal, ver Figura E.1, y será redireccionado a la página de inicio de sesión, siendo esta última igual a la Figura E.3. Seguidamente se deberán introducir las credenciales con las que el usuario se registró y tendrá acceso al sistema.

Recuperar contraseña

En caso de que el usuario necesite recuperar su contraseña, en la página de inicio de sesión, Figura E.3, puede hacer *click* en *Forgot your password?*, en donde se redirigirá a la página de recuperación de contraseña, igual a la Figura E.4.

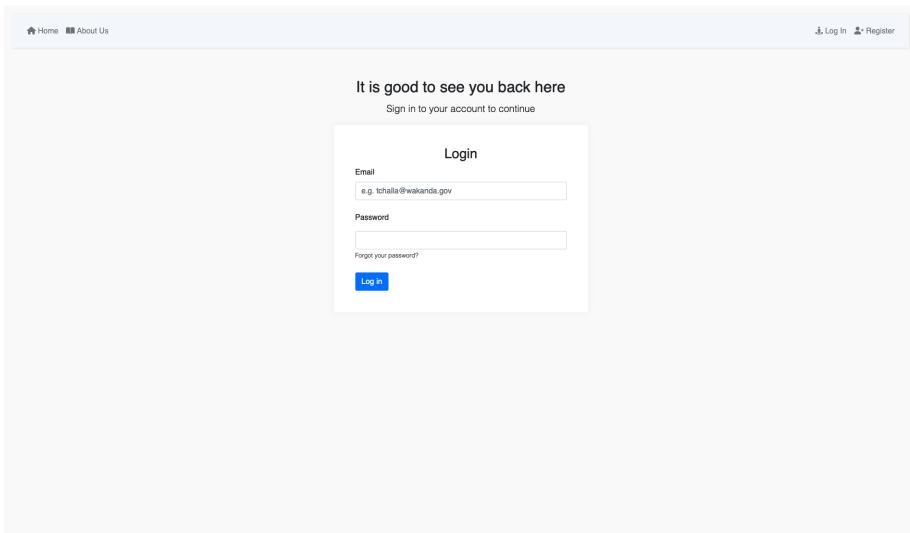


Figura E.3: Página de inicio de sesión.

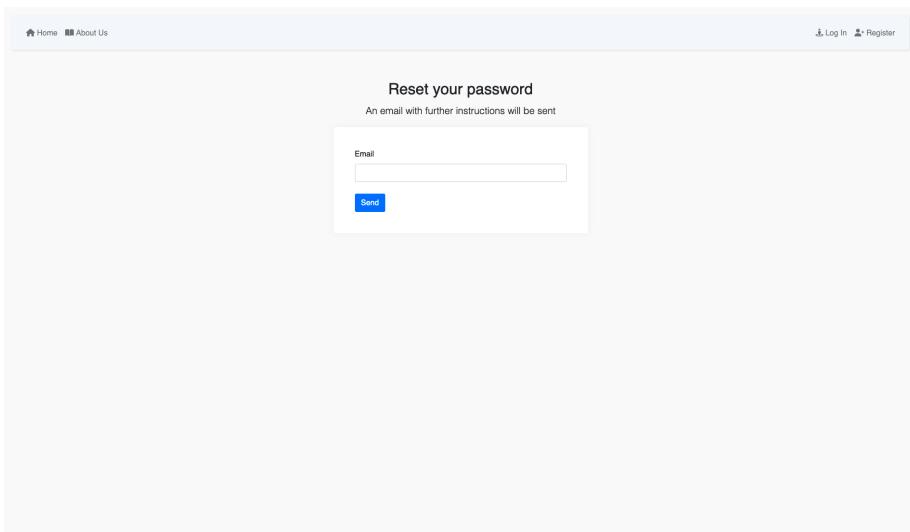


Figura E.4: Página de recuperar contraseña.

Introduciendo el correo electrónico y se le enviará un enlace al mismo para reestablecerla.

Crear un nuevo experimento

La funcionalidad principal proporcionada por UBUMLaas es la de crear experimentos de ML, para ello una vez se inicia sesión se llega al índice

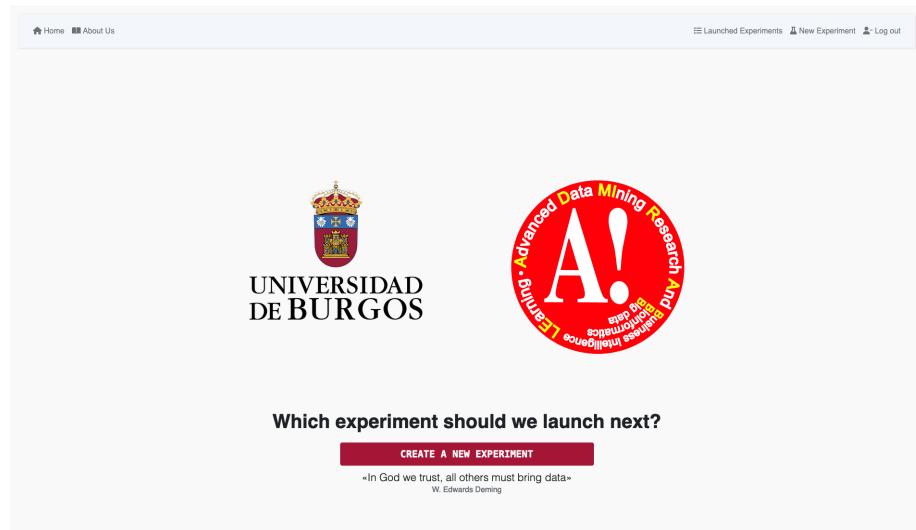


Figura E.5: Índice principal de UBUMLaas.

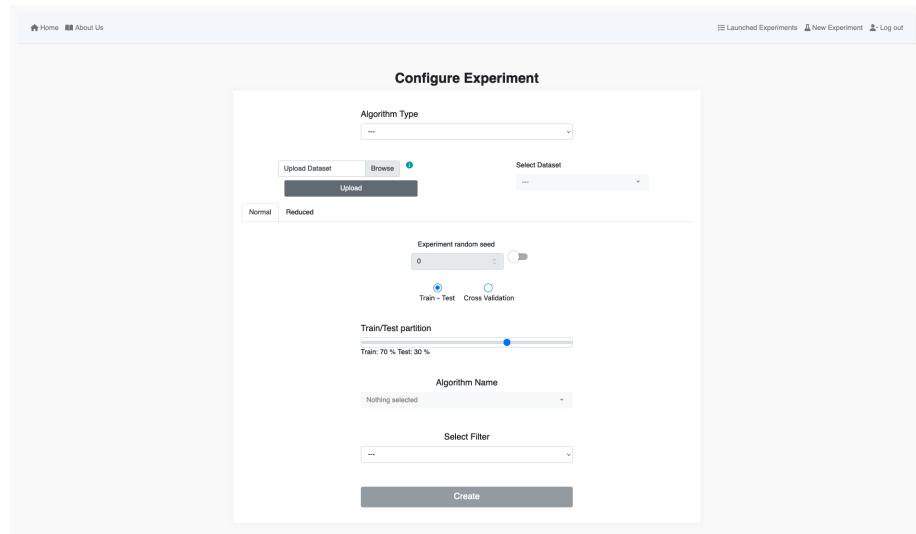


Figura E.6: Vista de crear experimento.

de la plataforma, tal y como aparece en la Figura E.5. Para llegar a la vista de crear un nuevo experimento se debe de hacer *click* o en el botón en mitad de la pantalla que indica *Create a new experiment*, o en la parte superior derecha, en el botón *New Experiment*. Cualquiera de los dos botones redirigirá al usuario a la vista deseada.

Llegando a una página similar a la Figura E.6 (la figura se encuentra con zoom al 90 % para poder visualizar toda la página).

En este momento el usuario podrá llenar todos los campos del formulario para crear su experimento deseado. Se recomienda llenar los campos en el siguiente orden:

1. Seleccionar le conjunto de datos a utilizar, pudiendo en este momento subir uno propio a la plataforma. Los nombres de los conjuntos de datos no son modificables luego asegúrese de que es correcto. En caso de subir uno propio la página se actualizará automáticamente en el momento en el que la carga haya sido satisfactoria. En caso de una visualización incompleta (diferente a la mostrada en la Figura E.7) se recomienda encarecidamente volver a acceder a la página y seleccionar el nuevo conjunto de datos desde la parte de conjuntos de datos existentes y disponibles, no siendo necesario volver a subirlo.
2. Seleccionar el tipo de algoritmo que se desea utilizar, disponiendo entre Clasificación, Regresión, Clasificación Semi-Supervisada, Multi-Clasificación, Clustering, o Mixed (Algoritmos compatibles con clasificación y regresión).
3. Seleccionar el algoritmo en concreto que desea utilizar.
4. Parametrizar el algoritmo tal y como se considere apropiado para el problema.
5. Seleccionar un filtro en caso de desear utilizarlo. Siendo estos filtros de selección de instancias.
6. Indicar una semilla en caso de considerarla necesaria su uso por motivos de reproducibilidad.
7. Indicar si se desea utilizar validación cruzada o partición en entrenamiento y pruebas. E indicación del número de *folds* o los porcentajes de partición, respectivamente.
8. Con todos los campos llenos. Se puede lanzar el experimento.

El resultado debería ser similar al representado por la Figura E.7, en el que se aprecia un experimento de clasificación con el conjunto de datos iris. Este experimento puede ser lanzado por un usuario según llega a la aplicación, puesto que todo lo que necesita se encuentra desde el minuto uno disponible.

Configure Experiment

Algorithm Type

Classification

Upload Dataset **Browse** **Upload**

Select Dataset
iris-classification.csv

Normal **Reduced**

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Class
Use	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Target	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
...
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Experiment random seed
0

Train - Test Cross Validation

Train/Test partition
Train: 70 % Test: 30 %

Algorithm Name

IBk

Knn(3)
Weight Inverse(false)
Weight Subtraction(false)

Select Filter

Create

Figura E.7: Formulario de crear experimento relleno.

Algorithm	Dataset	Start Date	Finish Date	State	View Result	Reuse	Delete
IBK	iris-classification.csv	27/04/2022 - 18:44:54	27/04/2022 - 18:45:00	Finalized	SEE	REUSE	DELETE
AdaBoost	iris-clustering.csv	14/04/2022 - 19:11:39	14/04/2022 - 19:11:40	Finalized	SEE	REUSE	DELETE
KNN	iris-classification.csv	14/04/2022 - 16:21:37	14/04/2022 - 16:21:38	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 16:06:03	14/04/2022 - 16:06:03	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 16:05:12	14/04/2022 - 16:05:13	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 16:01:37	14/04/2022 - 16:01:37	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 16:01:10	14/04/2022 - 16:01:10	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 15:57:56	14/04/2022 - 15:57:56	Finalized	SEE	REUSE	DELETE
KNN	cpu-regression.arff	14/04/2022 - 15:55:30	14/04/2022 - 15:55:30	Finalized	SEE	REUSE	DELETE

Figura E.8: Perfil del usuario.

Visualización de resultados

Llegado el momento como el lógico se querrá comprobar qué tan bien un modelo ha sido entrenado, y qué tal ha aproximado los resultados. Se disponen de dos formas de acceder a los resultados de un experimento en concreto

- Desde el perfil del propio usuario, tal y como se aprecia en la Figura E.8. Pulsando sobre el botón *See*.
- Desde el enlace recibido en un correo electrónico una vez que el experimento finalice.

Siguiendo con el experimento mostrado en la Figura E.7, en la Figura E.9 se pueden apreciar los resultados mostrados por la experimentación. Al haber sido un experimento sin validación cruzada, el sistema lo considera como si fuera una única *fold*, de ahí el $k=0$, en caso de usarse validación cruzada se tendrían $k=n$ en función de la n seleccionada.

- La matriz de confusión para cada una de las posibles etiquetas.
- El *AUC score*.
- El *F1 score*.

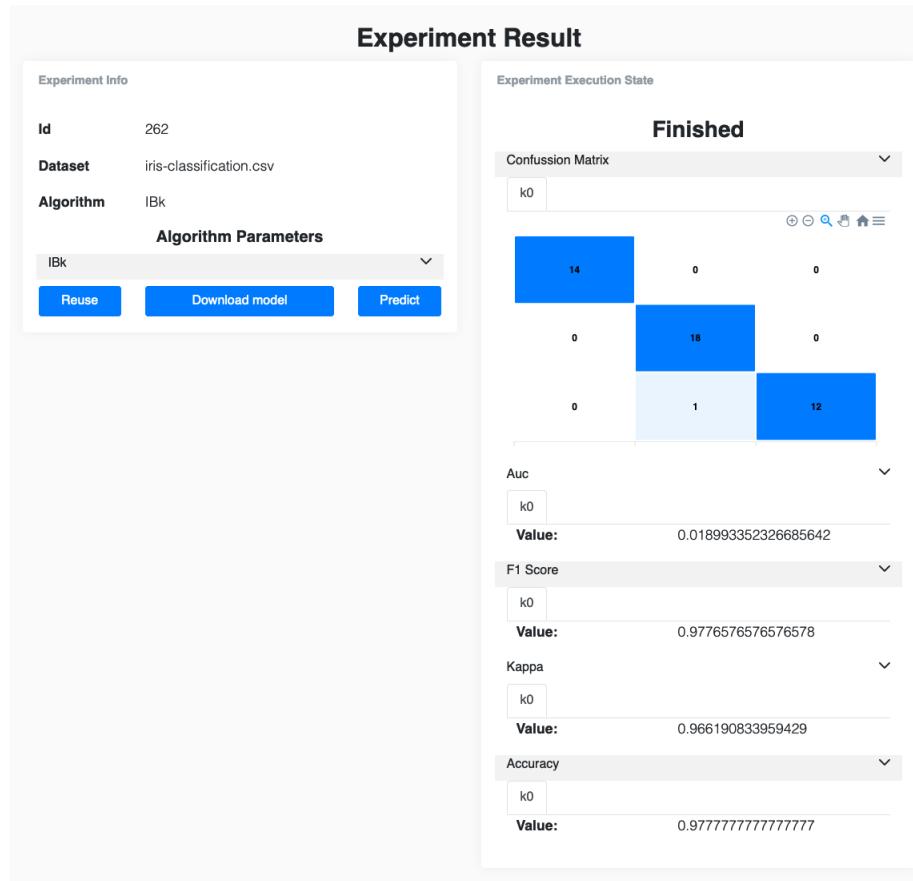


Figura E.9: Visualización de resultados.

- El *kappa score*.
- El *accuracy score*.

Se deja como trabajo del usuario el análisis de los diferentes valores y si tienen sentido para el problema que se plantea, no siendo representado su adecuación.

Predecir la clase correspondiente

Una vez que se posee un modelo entrenado, desde la página de visualización de los resultados del experimento, Figura E.9, se puede hacer *click* sobre el botón *Predict*, siendo inmediatamente redirigidos a la página de predicción de etiquetas.

Predict

Browse

Upload

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
...
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Predict

Figura E.10: Vista de antes de predecir.

Para poder predecir se deberá subir un conjunto de datos, el cual debe poseer **las mismas columnas (atributos), con exactamente los mismos nombres**, un ejemplo está en la Figura E.10, siguiendo el ejemplo con el que se viene trabajando.

Y posterior a la predicción se mostrarán los resultados tal y como cabría esperar, en la Figura E.11 se muestra el ejemplo, en caso de considerar que la predicción es correcta, se mostrará en verde, en este caso todas son rojas puesto que no se garantiza su corrección.

Predict					
	sepallength	sepalwidth	petallength	petalwidth	prediction_class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

Showing 1 to 10 of 150 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [15](#) Next

[Download](#)

Figura E.11: Vista de después de predecir.

Estadísticas de uso

Si se desean visualizar las estadísticas personales de uso de la aplicación, se debe acudir al perfil del usuario, el cuál es compartido con la lista de experimentos, desde la pantalla principal, Figura E.5, (o desde cualquier otra) se deberá pulsar en la parte superior derecha sobre el botón *Launched Experiments*, y se redirigirá al perfil.

En el perfil, Figura E.8, se pulsará sobre el botón verde debajo de la foto de perfil del usuario en el lado izquierdo, el botón muestra una cadena de texto en la que se indica *Statistics*. Seguidamente se abrirá una nueva card a la derecha, encima de todo lo que aparece, con las estadísticas del usuario.

Para cerrar la vista basta con hacer *click* de nuevo sobre el botón verde o sobre la cabecera de la **cart**.

En la Figura E.12 se muestra un ejemplo de las estadísticas de un usuario, las comentamos al detalle a continuación:

- Las cartas superiores son dos contadores, indican los experimentos **existentes** en la base de datos de la aplicación con identificador de usuario igual al usuario en cuestión. Y la segunda indica el número de conjunto de datos que el usuario posee en total, incluyendo los añadidos por defecto.
- El gráfico titulado *Experiments performed in the last 7 days*, tal y como la traducción referencia, muestra un gráfico con el número de experimentos que se han ejecutado por parte del usuario en los últimos 7 días, siendo el valor más a la derecha el día actual.
- El gráfico de tipo *pie* muestra el número de experimentos de cada tipo que el usuario ha ejecutado.
- El gráfico de barras permite conocer el tiempo en total que los experimentos de un usuario han estado en ejecución en el sistema, estando agrupados por tipos de algoritmos. Se puede cambiar la escala de tiempo para una mayor comodidad de interpretación.

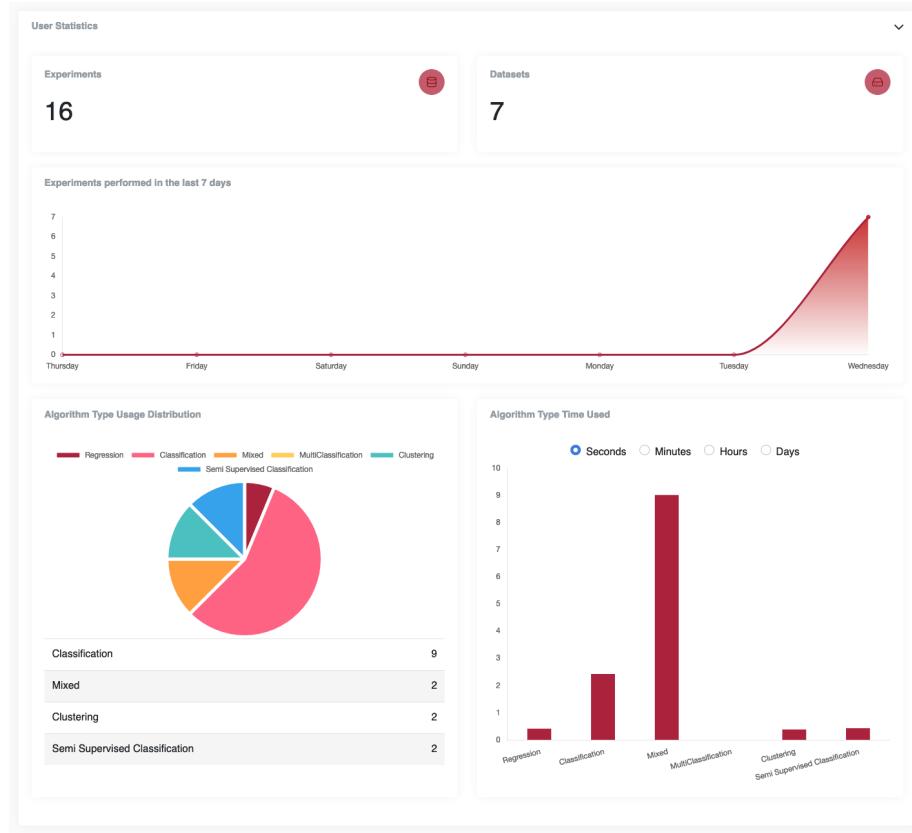


Figura E.12: Estadísticas de usuario.

Modificación de datos del usuario y actualizar contraseña

Todo usuario puede modificar sus datos personales, así como añadir una serie de datos que no son obligatorios. Para modificar los datos se realizará desde el perfil del usuario, haciendo *click* en el botón amarillo en la parte inferior izquierda, con la cadena de caracteres *Edit profile*.

En este momento se lanzará un modal el cual posee dos partes diferencias, modificación de datos personales, y en la parte inferior modificación de la contraseña.

Cuando se decida qué se quiere modificar, se deberá hacer *click* en el *checkbox* que se encuentra en la parte superior de cada formulario, lo cual habilitará la edición del mismo. Los formularios se encuentran en las Figuras E.13 y E.14, respectivamente.

Edit your data, Puente R X

Update user data

Username	Puente R
Email	p@p.es
Country	Monaco
Desired use	Research
Personal Website	www.example.com
Twitter username	@example
GitHub username	example
Institution	Universidad de Burgos
LinkedIn username	example98
Google Scholar URL	https://scholar.google.com/

Keep in mind that submitting this form will update ALL your data with the data on each field.

Update my data

Figura E.13: Formulario de edición de los datos de un usuario.

The screenshot shows a user interface for updating a password. At the top left is a checkbox labeled "Update Password". Below it are two input fields: one for "Password" and one for "Confirm Password", both represented by light gray rectangular boxes. At the bottom are two buttons: a dark gray "Close" button on the left and a blue "Update my password" button on the right.

Figura E.14: Formulario para cambiar la contraseña de un usuario.

Manual del administrador

A continuación, se detallan todas las funcionalidades añadidas que posee un administrador. Un administrador es un usuario en su base, por lo tanto, todas las funcionalidades descritas en la sección anterior también hacen referencia al administrador.

NavBar de administración

Tal y como se aprecia en la Figura E.15, el administrador posee una barra de navegación lateral en toda la aplicación, permitiendo acceder a esas funcionalidades en cualquier momento desde cualquier lugar. A su vez, en la parte superior derecha posee un acceso directo a la parte de administración, el botón *Administration*.

En caso de querer ocultar la barra lateral de administración, ya que no se va a utilizar en ese momento, en la parte superior izquierda aparece una X, haciendo *click* en ella se ocultará la barra lateral.

En las siguientes secciones se comentarán cada una de las opciones disponibles para los administradores.

Analytics Dashboard

Accediendo a través del botón *Dashboard* en el menú lateral, se accede a la vista de analíticas del sistema, en la que aparecen tanto estadísticas

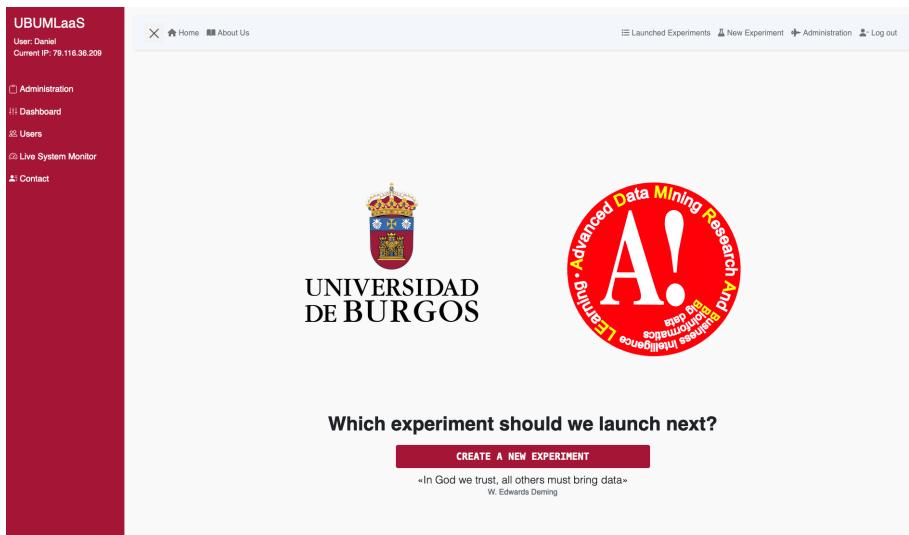
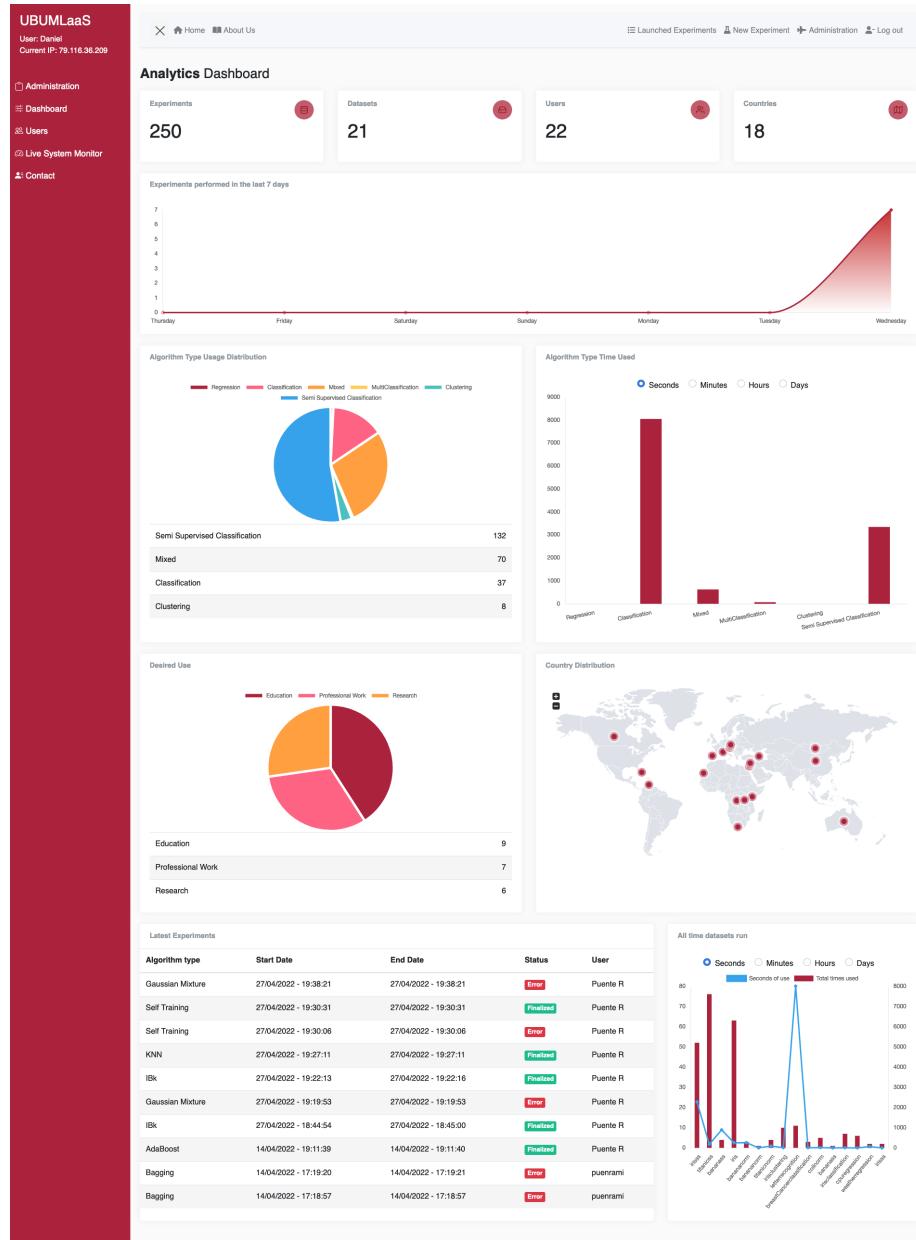


Figura E.15: Vista principal de administrador.

de los últimos 7 días, como globales del sistema, se puede apreciar en la Figura E.16.

Se dispone de la siguiente información:

- Cartas superiores
 - Número total de experimentos (los modelos) que se encuentran almacenados en el sistema.
 - Número total de conjuntos de datos distintos almacenados en el sistema.
 - Número de usuarios registrados en el sistema.
 - Número total de los países de los cuales los usuarios han dicho ser.
- *Experiments performed in the last 7 days.* Igual que para el usuario, pero con las estadísticas de todos los usuarios.
- *Algorithm Type Usage Distribution.* Estadísticas globales del número de experimentos de cada tipo que se han ejecutado.
- *Algorithm Type Time Used.* Distribución del tiempo de uso (ejecución) de los diferentes tipos de algoritmos.

Figura E.16: Vista de *Analytics Dashboard*.

The screenshot shows the 'Users' section of the UBUMLaas administration interface. On the left, a sidebar menu includes 'Administration', 'Dashboard', 'Users' (selected), 'Live System Monitor', and 'Contact'. The main area has a header with 'Home', 'About Us', 'Launched Experiments', 'New Experiment', 'Administration', and 'Log out'. A search bar and a 'New user' button are at the top right. Below is a table titled 'Users' with the following data:

#	Username	Email	Desired use	Country	Activated	Admin	Delete
1	Daniel	dpr1005@ali.ubu.es	Education	MN	<button>Deactivate</button>	<button>Remove Admin</button>	<button>Remove</button>
3	danix	danix@pafejo.com	Research	KE	<button>Deactivate</button>	<button>Make Admin</button>	<button>Remove</button>
4	Puente R	p@p.es	Education	MC	<button>Deactivate</button>	<button>Make Admin</button>	<button>Remove</button>
5	user-fake	user@userfake.user	Professional Work	ES	<button>Deactivate</button>	<button>Make Admin</button>	<button>Remove</button>
6	ntoolsecure	ntoolsecure@gmail.com	Education	AW	<button>Deactivate</button>	<button>Make Admin</button>	<button>Remove</button>
7	irene	irene@gmail.com	Education	ES	<button>Deactivate</button>	<button>Remove Admin</button>	<button>Remove</button>
8	juan	juan@ill.com	Professional Work	AM	<button>Activate</button>	<button>Make Admin</button>	<button>Remove</button>
9	o	o@o.co	Research	EH	<button>Activate</button>	<button>Make Admin</button>	<button>Remove</button>
10	manuek	m@m.co	Education	AW	<button>Activate</button>	<button>Make Admin</button>	<button>Remove</button>
11	fafafsfdsfsdf	prueba@admin.cadaf	Education	BS	<button>Activate</button>	<button>Make Admin</button>	<button>Remove</button>

Showing 1 to 10 of 22 rows | 10 rows per page | Page: 1 2 3 ...

Figura E.17: Vista de administración de usuarios.

- *Desired Use.* Estadísticas del uso que los usuarios han indicado que le van a dar primordialmente a la aplicación.
- *Country Distribution.* Representación de la ubicación geográfica de los usuarios. Siendo representado cada país por un único punto.
- *Latests Experiments.* Últimos 10 experimentos lanzados, pueden estar *In Progress*, o terminados, ya sea *Finalized* o bien *Error*. Mostrando la información mínima necesaria, así como el usuario dueño del experimento.
- *All Time Datasets Run.* Comparativa del tiempo de ejecución de algunos conjuntos de datos en comparación con el número de veces que han sido ejecutados. Se puede modificar la escala de tiempo.

Users

Se accede a la página de administración de usuarios a través del botón *Users* en la barra lateral. En este panel se pueden crear, (des)activar, dar/quitar privilegios de administración, o eliminar un usuario.

Tal y como se puede ver en la Figura E.17, se soporta la búsqueda por cualquiera de los campos que se visualizan, permitiendo encontrar a aquellos usuarios que interese en «un click».

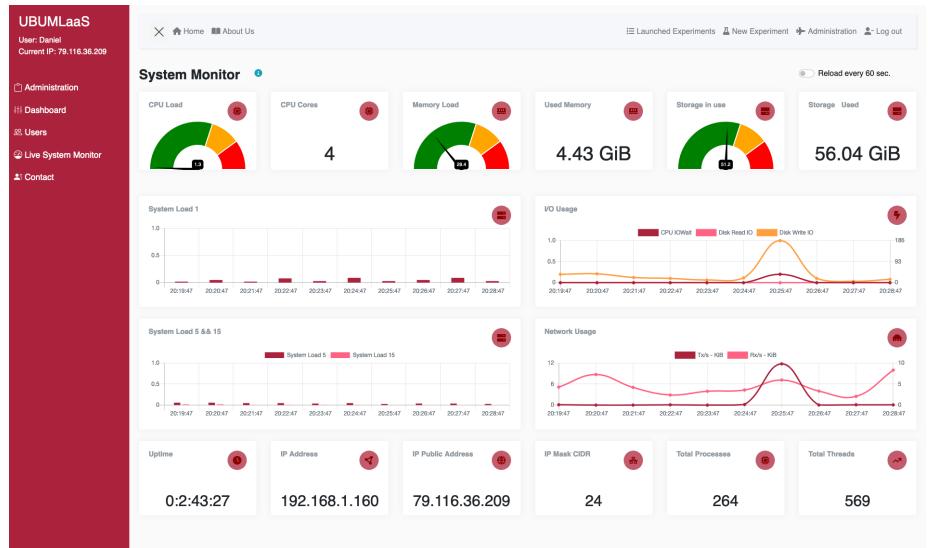


Figura E.18: Vista de *Live System Monitor*.

Un usuario no puede quitarse a sí mismo privilegios de administrador, ni desactivarse la cuenta, o eliminarla, teniendo que ser otro administrador el que lo haga; de esta manera el sistema siempre tendrá como mínimo un administrador.

A su vez se soporta crear un usuario haciendo *click* en el botón verde *New user*. Desplegándose un formulario y se deberán de llenar los campos de correo electrónico, nombre de usuario, país y uso que se va a hacer; las restricciones de los campos existentes deben de seguir cumpliéndose. Al usuario se le generará una contraseña y al correo electrónico llega un correo, valga la redundancia, de activación de la cuenta, pero deberá de re-establecer la contraseña como si la hubiera olvidado antes de poder iniciar sesión por primera vez.

Live System Monitor

A la monitorización del sistema en tiempo real se accede a través del botón *Live System Monitor* en la barra lateral. Cuando se hace *click* se redirecciona a una vista similar a la que aparece en la Figura E.18. En la parte superior derecha de la página a la que se llega hay un botón para activar si se quiere que la página se auto-recargue cada 60 segundos, desde el momento en el que se hace *click*.

NOTA. Es importante tener en cuenta que esta pantalla se ha diseñado para monitores de más de 23 pulgadas, por lo que su visualización en monitores de menor tamaño puede no ser óptima o encontrar ciertos solapamientos. En la Figura E.18 se aprecia la disposición correcta de todos los componentes.

La información que se muestra es la siguiente (todas las unidades que se muestran son calculadas dinámicamente, seleccionando la mayor disponible):

- Las cartas superiores muestran, de izquierda a derecha:
 - *CPU Load.* La carga de la CPU porcentualmente.
 - *CPU Cores.* El número total de núcleos del sistema.
 - *Memory Load.* El uso total de la memoria en forma de gráfico.
 - *Used Memory.* El valor total de memoria en uso en el sistema.
 - *Storage in use.* Almacenamiento total del sistema en vista de gráfico.
 - *Storage Used.* El valor total de almacenamiento en uso.
- *System Load 1/5/15.* Cada gráfico representa la carga media del sistema en los últimos 1, 5 y 15 minutos, respectivamente.
- *I/O Usage.* Interrupciones de tipo *Input/Output* de la CPU y del disco.
- *Network Usage.* Tamaño total de información transmitida y recibida en el periodo de tiempo.
- Las cartas inferiores muestran, de izquierda a derecha:
 - *Uptime.* Tiempo total desde que el sistema se inició. Formato: días:horas:minutos:segundos.
 - *IP Address.* Dirección IP del equipo/servidor donde se encuentra la plataforma desplegada.
 - *IP Public Address.* Dirección IP pública del sistema.
 - *IP Mask CIDR.* Máscara de subred en formato CIDR.
 - *Total Processes.* Número total de procesos que existen en el sistema en ejecución.
 - *Total Threads.* Número total de hilos en el sistema.

E.3. IS-SSL

A continuación, se presenta el manual de usuario de las bibliotecas de algoritmos desarrolladas. Permitiendo a cualquier usuario comprender IS-SSL y poder hacer uso de estas.

Requisitos de usuarios

Los requisitos mínimos para poder hacer uso de IS-SSL son:

- Tener instalado Python 3.7+.
- Tener instalado y configurado PIP o Conda.
- Disponer de un editor de textos.
- Tener instaladas las bibliotecas necesarias para su correcto funcionamiento.

Instalación

Por comodidad para el usuario, IS-SSL se ha dividido en dos bibliotecas, una formada por los algoritmos de selección de instancias, y una segunda por aquellos algoritmos de aprendizaje semi-supervisado.

El proceso de instalación de cualquiera de las dos bibliotecas es muy sencillo, siendo integrable en cualquier fichero de requerimientos, ya sea para PIP o Conda.

Las dos bibliotecas se encuentran publicadas en PyPI² desde su versión 1.0, la cual fue una primera versión alpha estable con los primeros algoritmos publicados. La versión 3.3 es la versión estable (la final) que se ha publicado.

Para realizar la instalación se deben seguir los siguientes pasos para cualquier LIB, $\text{LIB} \in \{\text{IS-DNX}, \text{SSL-DNX}\}$.

1. Acceder a PyPi, desde [5].
2. Introducir en el campo de búsqueda «LIB».
3. Seleccionar la biblioteca correspondiente de entre la lista mostrada.

²Python Package Index es un repositorio de *software* para el lenguaje de programación de Python.

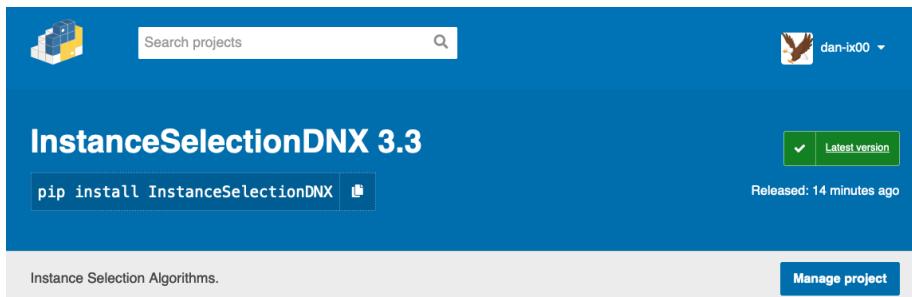


Figura E.19: Vista de la biblioteca de algoritmos de selección de instancias en PyPI.

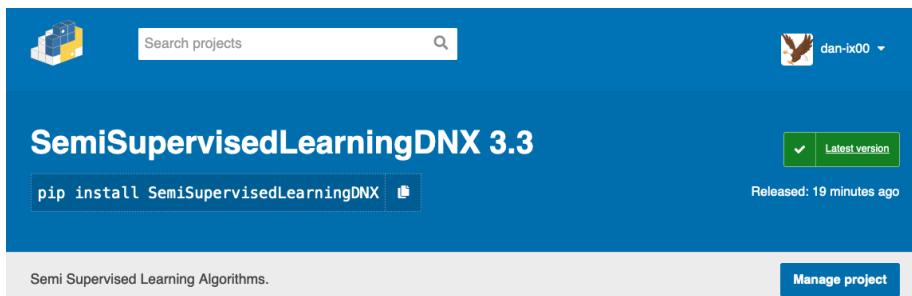
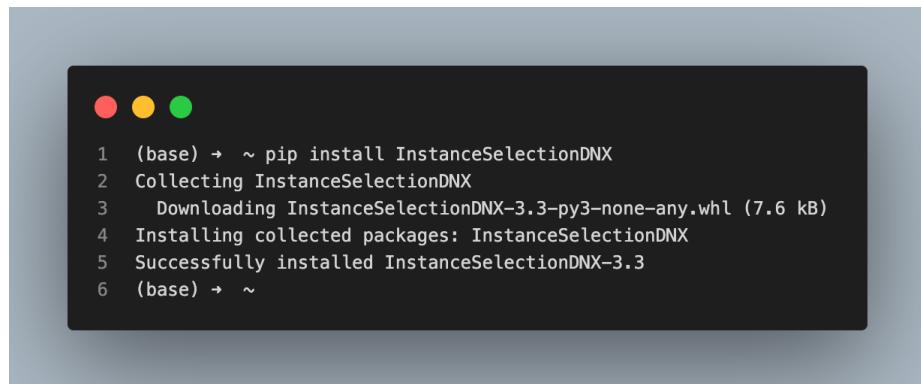


Figura E.20: Vista de la biblioteca de algoritmos de aprendizaje semi-supervisado en PyPI.

4. Copiar el comando de instalación.
5. Abrir una terminal con soporte a Python y PIP.
6. Introducir el comando copiado.
7. En caso de que se nos pregunte si se quiere proceder con la descarga, indicar que sí con una S en caso de que esté en español, o con Y en el caso inglés/internacional.
8. Cuando finaliza la instalación, la biblioteca se encontrará lista para su uso.

Manual del usuario

A continuación, se documentan las funcionalidades de las bibliotecas, desde su importación, a uso y especificación de los diferentes parámetros

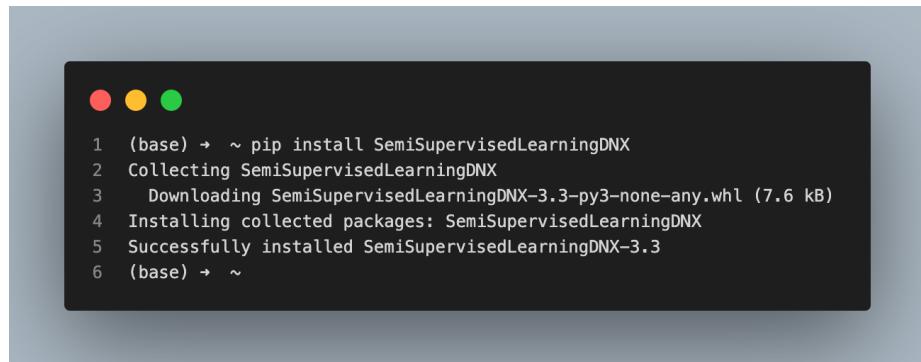


```

(base) → ~ pip install InstanceSelectionDNX
Collecting InstanceSelectionDNX
  Downloading InstanceSelectionDNX-3.3-py3-none-any.whl (7.6 kB)
Installing collected packages: InstanceSelectionDNX
Successfully installed InstanceSelectionDNX-3.3
(base) → ~

```

Figura E.21: Instalación de la biblioteca de selección de instancias.



```

(base) → ~ pip install SemiSupervisedLearningDNX
Collecting SemiSupervisedLearningDNX
  Downloading SemiSupervisedLearningDNX-3.3-py3-none-any.whl (7.6 kB)
Installing collected packages: SemiSupervisedLearningDNX
Successfully installed SemiSupervisedLearningDNX-3.3
(base) → ~

```

Figura E.22: Instalación de la biblioteca de semi-supervisado.

de entrada y salida esperados. A modo de resumen se puede destacar que todos los algoritmos siguen la misma estructura interna luego el aprendizaje y familiarización es relativamente rápido.

Biblioteca de algoritmos de selección de instancias

Importar

Para poder trabajar con los algoritmos de selección de instancias se deben de importar en el fichero en el que se quieran utilizar. Para ello se importan como cualquier otro paquete de Python, supongamos que queremos utilizar el algoritmo ENN, lo importaremos de la siguiente manera:

```
from InstanceSelectionDNX import ENN
```

De esta forma podemos sustituir ENN por el algoritmo que deseemos de entre los disponibles y tenerlo a nuestra disposición para su uso.

Todos los algoritmos están codificados como `class` por lo tanto se debe de instanciar antes de poder hacer uso de este.

Uso

Como se ha comentado al comienzo, todos los algoritmos poseen la misma estructura. Todos ellos poseen el método `filter` de tal manera que una vez se haya instanciado se podrá llamar al método y se obtendrá como resultado el conjunto de datos reducido.

Todos los algoritmos en su instanciación reciben aquellos parámetros que son necesarios para la configuración y su uso posterior, mientras que cuando se realiza el filtrado únicamente reciben el conjunto de datos dividido, por un lado, los atributos y por otro lado la clase.

Tanto las entradas como las salidas deben ser objetos de tipo `DataFrame` de `Pandas`.

```
from InstanceSelectionDNX import ENN
import pandas as pd

data = pd.DataFrame([[1, 2, 3, 4],
                     [4, 3, 2, 1],
                     [1, 2, 4, 3],
                     [2, 1, 3, 4]])
target = pd.DataFrame([0, 0, 1, 1])

model = ENN(nearest_neighbors=3, power_parameter=2)

data_red, label_red = model.filter(data, target)
```

Listing E.1: Ejemplo de uso de ENN

Biblioteca de algoritmos de aprendizaje semi-supervisado

Importar

De manera análoga a la otra biblioteca, importaremos el paquete y seleccionaremos cuál es el algoritmo que se desea utilizar, por ejemplo:

```
from SemiSupervisedLearningDNX import TriTraining
```

Pudiendo sustituir `TriTraining` por el algoritmo deseado.

Todos los algoritmos están codificados como `class` por lo tanto se debe de instanciar antes de poder hacer uso de este.

Uso

Los algoritmos siguen la misma estructura interna que los propios de

Scikit-Learn, por lo que una vez instanciados (con sus respectivos parámetros de configuración) bastará con llamar al método `fit` de cada uno de ellos, así como para predecir al método correspondiente, denominado `predict`.

- **fit:** recibe como argumentos dos parámetros, las instancias y las etiquetas o clases, siendo -1 aquellas que se desconozcan y se quieran utilizar para entrenar el algoritmo.
- **predict:** recibe únicamente las instancias que se quieren etiquetar. Devuelve estas instancias etiquetadas.

```

from SemiSupervisedLearningDNX import TriTraining
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris

model = TriTraining(
    random_state = 42,
    c1 = GaussianNB, c1_params = None,
    c2 = KNeighborsClassifier, c2_params = {n_neighbors: 2}
)

iris = load_iris()
X = iris['data']
y = iris['target']

X = pd.DataFrame(X), y = pd.DataFrame(y)

val = [True if i % 2 == 0 else False for i in range(len(y))]
y.loc[val] = -1

X, X_test, y, y_test = train_test_split(X.to_numpy(),
                                         y.to_numpy())

X = pd.DataFrame(X), y = pd.DataFrame(y)

model.fit(X, y)
y_pred = model.predict(X_test)

```

Listing E.2: Ejemplo de uso de IS-SSL

Ejemplo de uso combinado de ambas bibliotecas

```

from SemiSupervisedLearningDNX import TriTraining
from InstanceSelectionDNX import ENN
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris

if __name__ == "__main__":
    model = TriTraining(
        random_state = 42,
        c1 = GaussianNB, c1_params = None,
        c2 = KNeighborsClassifier, c2_params = {n_neighbors: 2},
        c3 = DecisionTreeClassifier, c3_params = None
    )
    filter_model = ENN(nearest_neighbors = 5, power_parameter = 2)

    iris = load_iris()
    X = iris['data']
    y = iris['target']

    X = pd.DataFrame(X)
    y = pd.DataFrame(y)
    X, y = filter_model.filter(X, y)

    val = [True if i % 2 == 0 else False for i in range(len(y))]
    y[val] = -1

    X, X_test, y, y_test = train_test_split(X.to_numpy(),
                                             y.to_numpy())

    X = pd.DataFrame(X)
    y = pd.DataFrame(y)

    model.fit(X, y)
    y_pred = model.predict(X_test)
    print(accuracy_score(y_true=y_test, y_pred=y_pred))

```

Listing E.3: Ejemplo de uso de IS-SSL

Bibliografía

- [1] Creative commons - public domain. <https://wiki.creativecommons.org/wiki/CC0>.
- [2] Creative commons - public domain. <https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es>.
- [3] Pycharm. <https://www.jetbrains.com/pycharm/>.
- [4] Python download. <https://www.python.org/getit/>.
- [5] Python package index. <https://pypi.org/>.
- [6] Remote ssh. <https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh>.
- [7] Salario medio investigador españa. <https://es.indeed.com/career/investigador/salaries>.
- [8] Salario medio programador junior españa. <https://es.indeed.com/career/programador-junior/salaries>.
- [9] Scikit-learn nearest neighbors. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html>.
- [10] Travis-ci ubumlaas. <https://app.travis-ci.com/github/dpr1005/UBUMLaaS>.
- [11] Visual studio code. <https://code.visualstudio.com/>.
- [12] Scrum (desarrollo de software), 2022. [https://es.wikipedia.org/wiki/Scrum-\(desarrollo-de-software\)](https://es.wikipedia.org/wiki/Scrum-(desarrollo-de-software)).

- [13] Ricardo Barandela, Francesc J Ferri, and J Salvador Sánchez. Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(06):787–806, 2005.
- [14] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.
- [15] BOE. Disposición adicional cuarta. tarifa de primas para la cotización a la seguridad social por accidentes de trabajo y enfermedades profesionales., 2019.
- [16] BOE. Orden pcm/244/2022, de 30 de marzo, por la que se desarrollan las normas legales de cotización a la seguridad social, desempleo, protección por cese de actividad, fondo de garantía salarial y formación profesional para el ejercicio 2022, 2022.
- [17] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.
- [18] H Frank Cervone. Understanding agile project management methods using scrum. *OCLC Systems & Services: International digital library perspectives*, 2011.
- [19] Ministerio de Empleo y Seguridad Social. Bases y tipos de cotización 2022.
- [20] Sjoerd de Vries and Dirk Thierens. A reliable ensemble based approach to semi-supervised learning. *Knowledge-Based Systems*, 215:106738, 2021.
- [21] Geoffrey Gates. The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433, 1972.
- [22] Marek Grochowski and Norbert Jankowski. Comparison of instance selection algorithms ii. results and comments. In *International Conference on Artificial Intelligence and Soft Computing*, pages 580–585. Springer, 2004.
- [23] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

- [24] IEEE. Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, pages 1–40, 1998.
- [25] IEEE. Systems and software engineering—software life cycle processes—part 2: Relation and mapping between iso/iec/ieee 12207:2017 and iso/iec 12207:2008. *IEEE Std 12207-2-2020*, 2020.
- [26] ISTR Ingeniería Software y Tiempo Real. Ieee830-esp - ctr.unican.es, 2020.
- [27] Norbert Jankowski and Marek Grochowski. Comparison of instances selection algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, pages 598–603. Springer, 2004.
- [28] Enrique Leyva, Antonio González, and Raúl Pérez. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 48(4):1523–1537, 2015.
- [29] Junnan Li, Qingsheng Zhu, and Quanwang Wu. A self-training method based on density peaks and an extended parameter-free local noise filter for k nearest neighbor. *Knowledge-Based Systems*, 184:104895, 2019.
- [30] Huan Liu and Hiroshi Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115, 2002.
- [31] Microsoft. Remote development tips and tricks, 2022.
- [32] Dan Radigan. El backlog del producto: la lista de tareas pendientes definitiva, 2021.
- [33] Julio Roche. Scrum: roles y responsabilidades, 2020.
- [34] Synk. What is a software license?
- [35] Isaac Triguero, José A Sáez, Julián Luengo, Salvador García, and Francisco Herrera. On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification. *Neurocomputing*, 132:30–41, 2014.
- [36] European Union. Jla - find and compare software licenses.
- [37] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

- [38] Di Wu, Mingsheng Shang, Xin Luo, Ji Xu, Huyong Yan, Weihui Deng, and Guoyin Wang. Self-training semi-supervised classification based on density peaks of data. *Neurocomputing*, 275:180–191, 2018.
- [39] Di Wu, Mingsheng Shang, Xin Luo, Ji Xu, Huyong Yan, Weihui Deng, and Guoyin Wang. Self-training semi-supervised classification based on density peaks of data. *Neurocomputing*, 275:180–191, 2018.
- [40] Yan Zhou and Sally Goldman. Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602. IEEE, 2004.
- [41] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541, 2005.



Esta obra está bajo una licencia Creative Commons Atribución - No Comercial - Sin Derivadas - 4.0 Internacional ([CC BY-NC-ND 4.0](#)).