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 Documentación Técnica

Presentado por Daniel Puente Ramírez
en Universidad de Burgos — 17 de abril de 2022
Tutor: Dr. Álvar Arnaiz González

# Índice general

# Índice de figuras

# Índice de tablas

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. Desde el punto de vista de la temporalidad y la viabilidad. Es una parte fundamental del ya que permitirá visualizar el escenario en el que se desarrollará el proyecto, permitiendo hacer una alineación estratégica de todos los elementos que se deben completar para finalizar correctamente el proyecto.

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 u *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 aproximadamente.

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 multiples 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 librerías 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 [12], 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

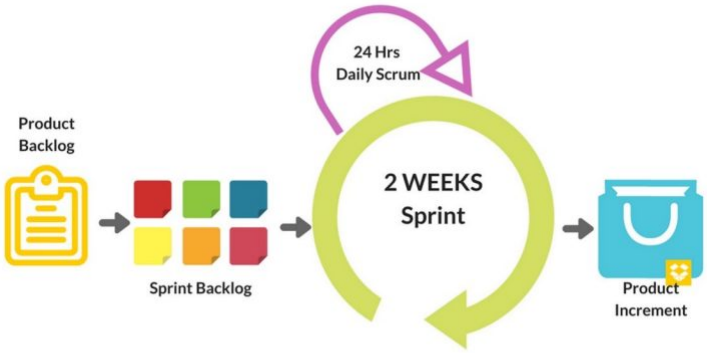


Figura A.1: Metodología *scrum*.

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 periodos 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 perfectaemnte descritas antes de asignarlas esta periodicidad, implicnddo 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 en *backlog* [22].

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* [23].

- ***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 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 tiene que 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 encargados 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* [25], *Co-Training* [10], *Tri-Training* [28] y *Democratic Co-Learning* [27].

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 veintisiete de octubre y el cinco de noviembre, de dos mil veintiuno.

### *Sprint* 1: Chad

- ***Planning meeting***

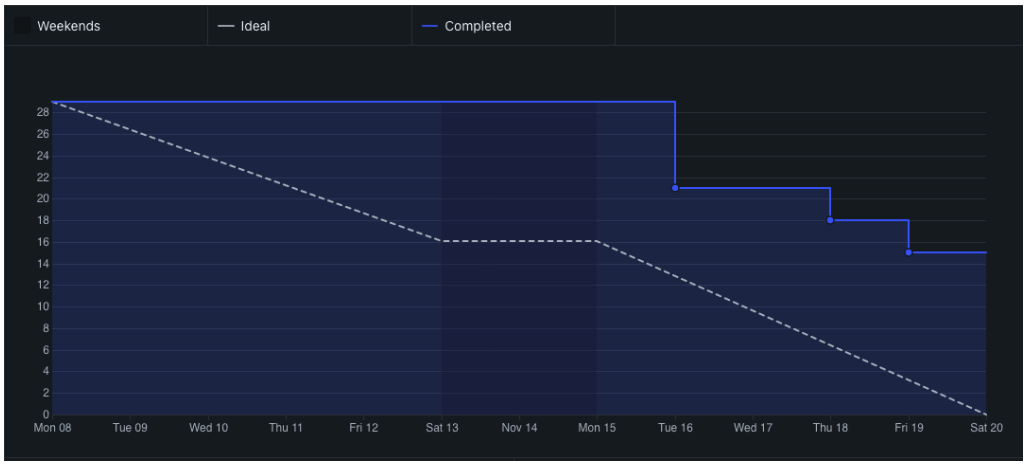


Figura A.2: *Burndown Chart Sprint 1.*

Objetivos del primer *sprint*:

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* [20], *Comparison of instances seletion algorithms I. algorithms survey* [17] y *Comparison of instance selection algorithms II. Results and comments* [14].
3. Implementación de las técnicas de reducción del conjunto de entrenamiento, basados en k-NN.

- **Marcas temporales** El *sprint* se desarrolla entre el ocho y el diecinueve de noviembre de dos mil veintiuno. 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*.

  El *sprint* se desarrolla entre el veintidós de noviembre y el tres de diciembre de dos mil veintiuno. Han sido dedicadas al desarrollo del proyecto treinta y ocho horas.

- ***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* [15], retrasando el trabajo cuatro horas, entre identificación y re-programación. Este error se descubrió mientras se investigaba otro error, en este caso el algoritmo *Iterative Case Filtering, ICF* [11] terminaba en error buscando los k-NN de las últimas instancias.

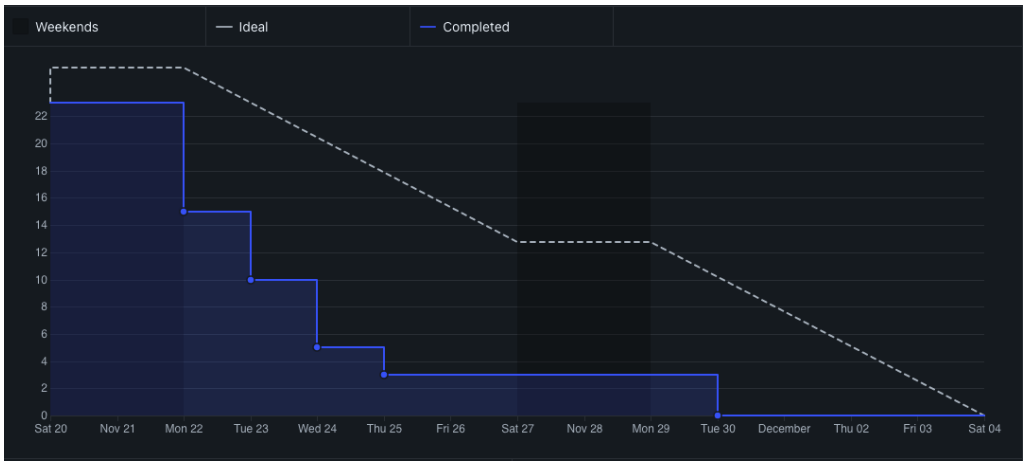


Figura A.3: *Burndown Chart Sprint 2.*

La implementación de los algoritmos *Reduced Nearest Neighbor, RNN* [13] y *Modified Selective Subset, MSS* [9] ha sido relativamente asequible 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 LaTeXlo 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 asíntotas donde ya no mejora la clasificación.
  4. Validación de los algoritmos de selección de instancias con *Weka* y *KNN*.

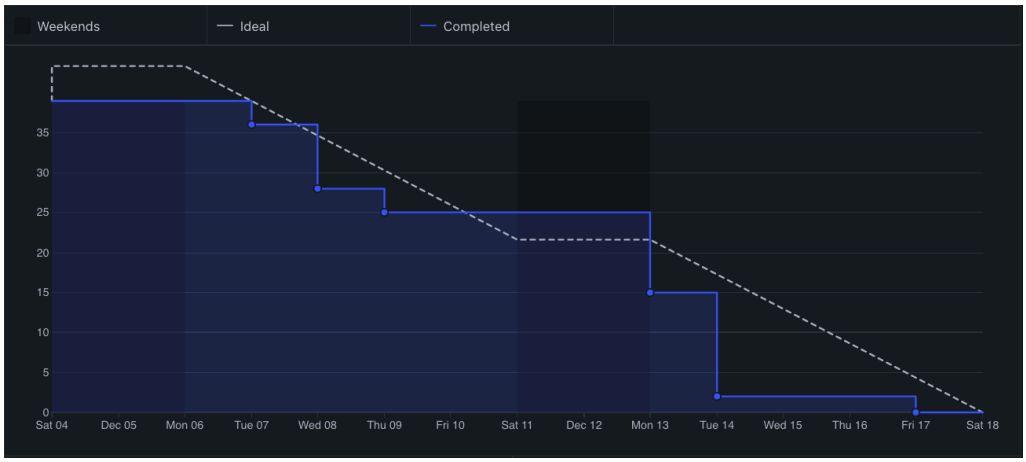


Figura A.4: *Burndown Chart Sprint 3.*

- **Marcas temporales**

  El *sprint* se desarrolla entre el seis y el diecisiete de diciembre de dos mil veintiuno. Han sido dedicadas al desarrollo del proyecto 45 horas.

- ***Burndown chart***

  El trabajo realizado 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 ±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

### ***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 dieciocho y el veintitrés de diciembre de dos mil veintiuno. 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*** Ha 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.

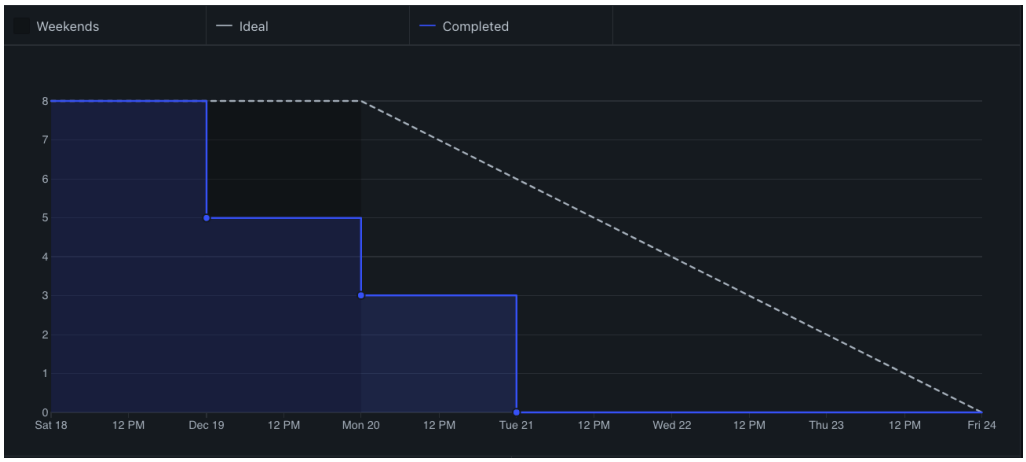


Figura A.5: *Burndown Chart Sprint 4.*

### *Sprint* 5: Murph

- ***Planning meeting***

  Objetivos del quinto *sprint*:

  1. Codificación de los algoritmos de aprendizaje semi-supervisado: Co-Training [10], Tri-Training [28], Democratic Co-Learning [27] Y del algoritmo de selección de instancias DROP3 [24].
  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 veinticuatro de diciembre de dos mil veintiuno y el diez de enero de dos mil veintidós. Tiene una duración igual a las festividades navideñas.

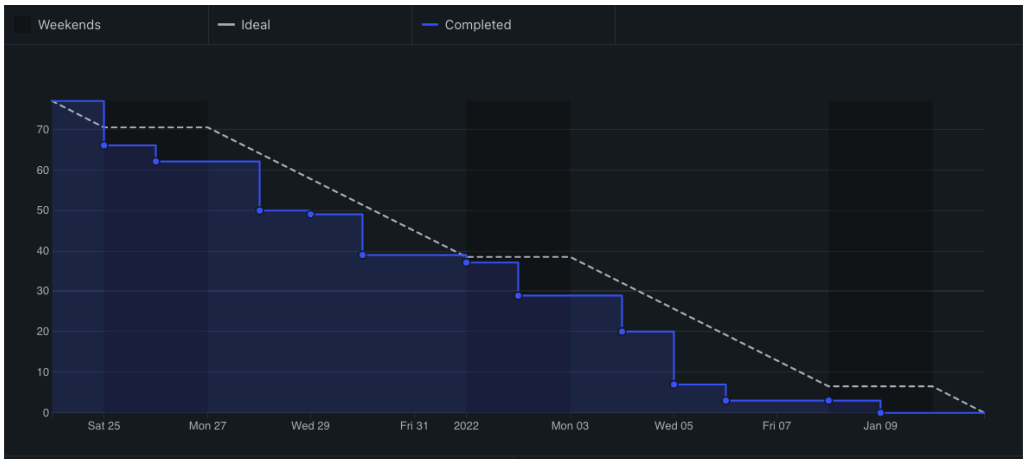


Figura A.6: *Burndown Chart Sprint 5.*

- ***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.

### *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 once de enero de dos mil veintidós, y finaliza el lunes diecisiete de enero de dos mil veintidós.

- ***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 librería *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 a lo que se comentó en la anterior reunión. Si bien ha sido un *sprint* más enfocado a «cerrar» una 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.

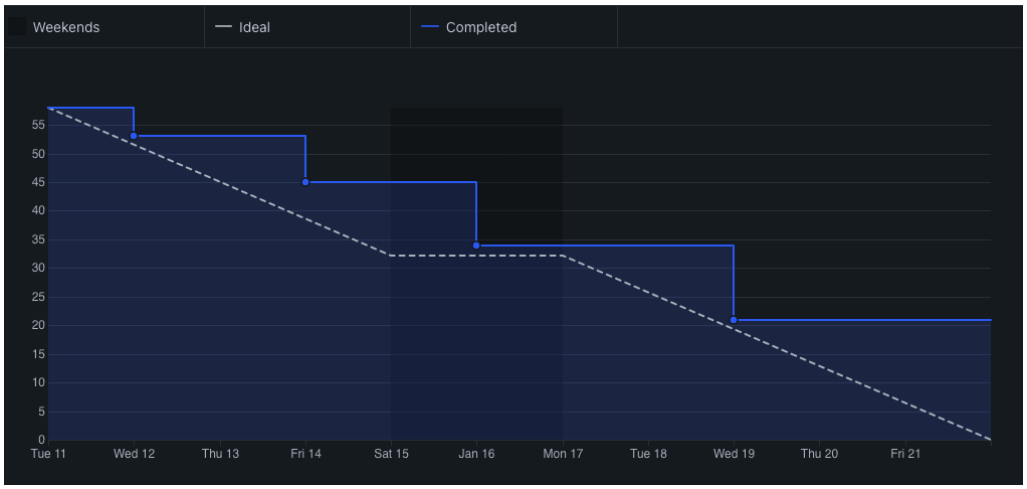


Figura A.7: *Burndown Chart Sprint 6.*

### *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 veinticinco de enero de dos mil veintidós y el dos de febrero de dos mil veintidós. Es un *sprint* muy rápido de preparación para poder comenzar con la parte de UBUMLaaS.

- ***Burndown chart***

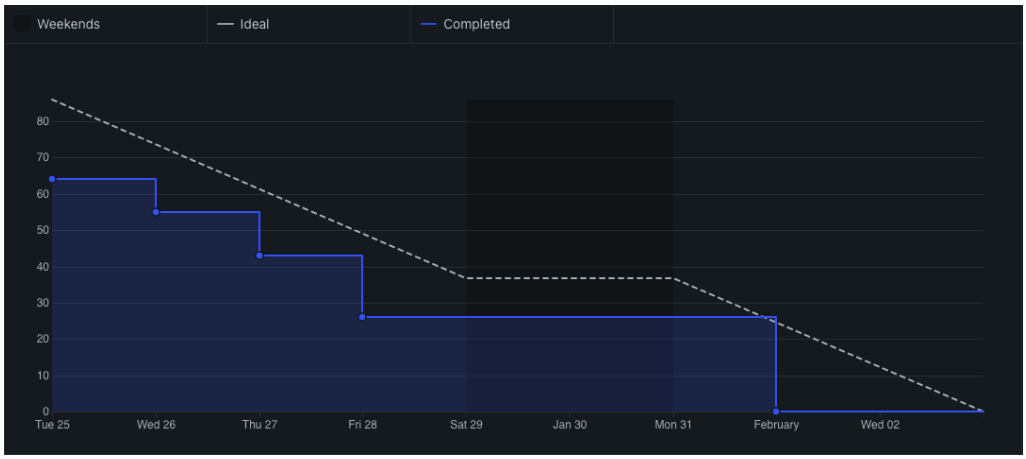


Figura A.8: *Burndown Chart Sprint 7.*

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 del mismo, pero sí fueron cerradas previo el inicio de este nuevo *sprint*, el gráfico 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 temporarización 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 UBUMLaaS para los algoritmos de aprendizaje Semi-Supervisado.
  2. Integrar los algoritmos implementados de Semi-Supervisado en la plataforma UBUMLaaS.

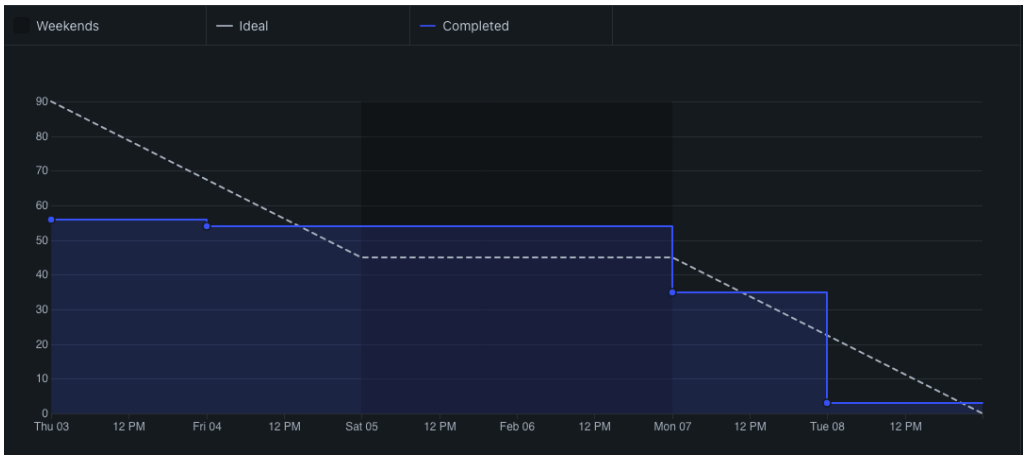


Figura A.9: *Burndown Chart Sprint 8.*

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.)
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 PIP[1].

- **Marcas temporales**

  Este *sprint* se desarrolla entre el tres de febrero de dos mil veintidós y el ocho de febrero de dos mil veintidós. 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

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

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 planificación 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 nueve de febrero de dos mil veintidós y el dieciséis de febrero de dos mil veintidós.

- ***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


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

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.

- ***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 re-programado 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 Jose Luis. (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.

### ***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 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 dieciséis de febrero de dos mil veintidós y el veintiuno de febrero de dos mil veintidós.

- ***Burndown chart***

  Este *sprint* ha sido más ajustado el número de horas invertidas en el desarrollo de las tareas marcadas en comparación 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

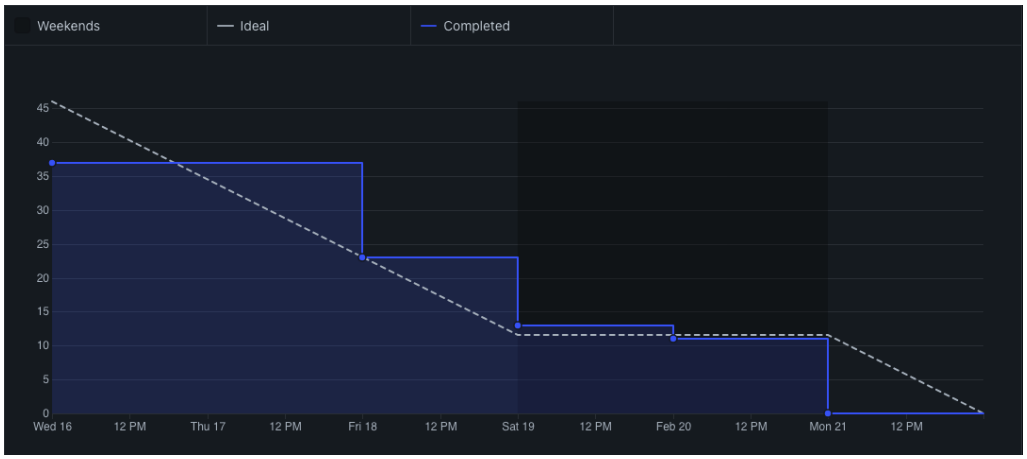


Figura A.11: *Burndown Chart Sprint 10.*

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 librarías.

### *Sprint* 11: Nutts

- ***Planning meeting***

  Objetivos del undécimo *sprint*:

  1. Con [19] 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 librería $IS\_SSL$ dentro de CI/CD [2]
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, RRSS, ...)
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 veintidós de febrero de dos mil veintidós y el uno de marzo de dos mil veintidós.

- ***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

[2] Método 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.

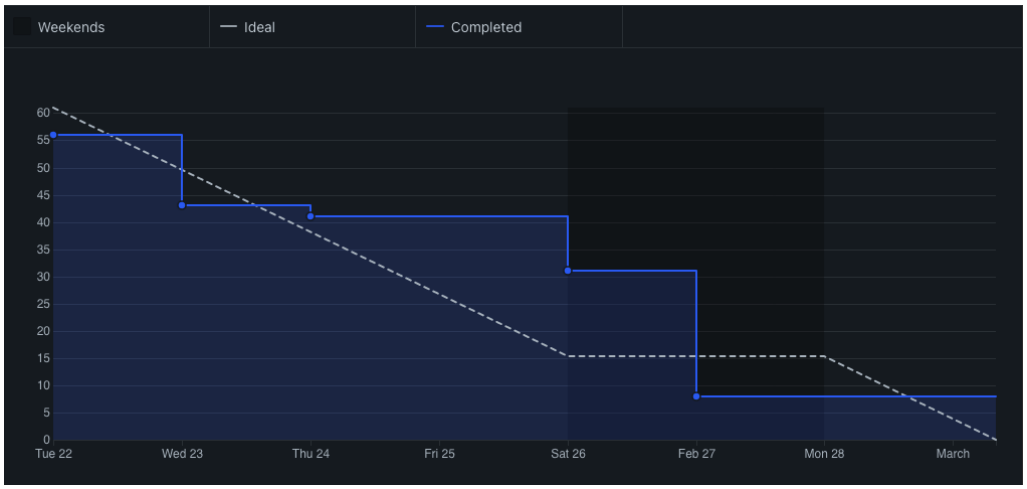


Figura A.12: *Burndown Chart Sprint 11.*

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.

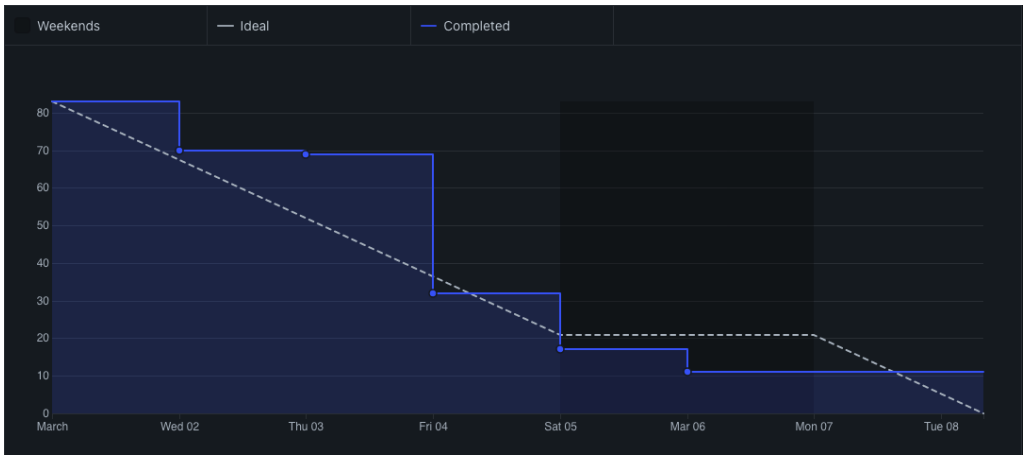


Figura A.13: *Burndown Chart Sprint 12.*

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 [26].

- **Marcas temporales**

  Este *sprint* se desarrolla entre el uno de marzo de dos mil veintidós y el ocho de marzo de dos mil veintidós.

- ***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 ha 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 que no «llegaba» a todo lo planificado, las reuniones llegan a un punto en el cuál 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 [19].
  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 a sí como los cambios en la interfaz.
  9. Revisar comentarios hechos por Alvar en la memoria.

- **Marcas temporales**

  Este *sprint* se desarrolla entre el ocho de marzo de dos mil veintidós y el quince de marzo de dos mil veintidós.

- ***Burndown chart***

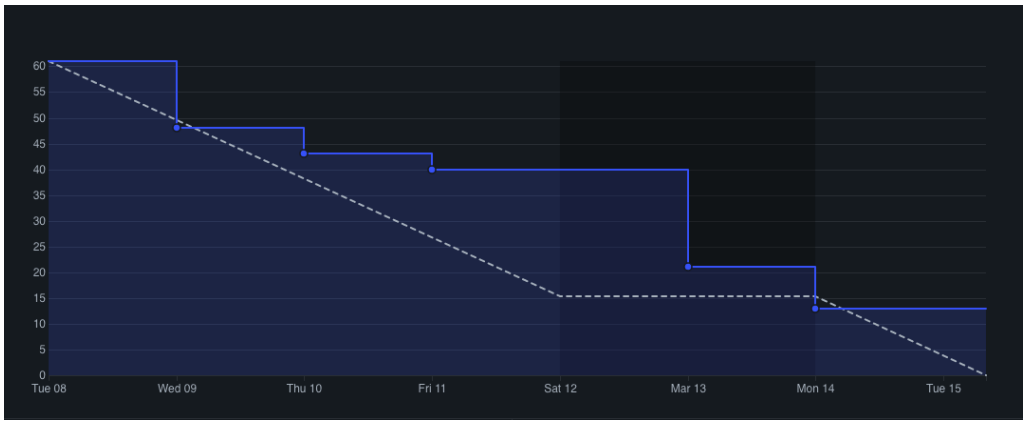


Figura A.14: *Burndown Chart Sprint 13.*

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 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».


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

3. Realizar la experimentación con picos de densidad y ruido.
4. Implementación de los algoritmos LSSm y LSBo, ver [18].

- **Marcas temporales**

  Este *sprint* se desarrolla entre el quince de marzo de dos mil veintidós y el veintidós de marzo de dos mil veintidós.

- ***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.

- ***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 corazonada 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.

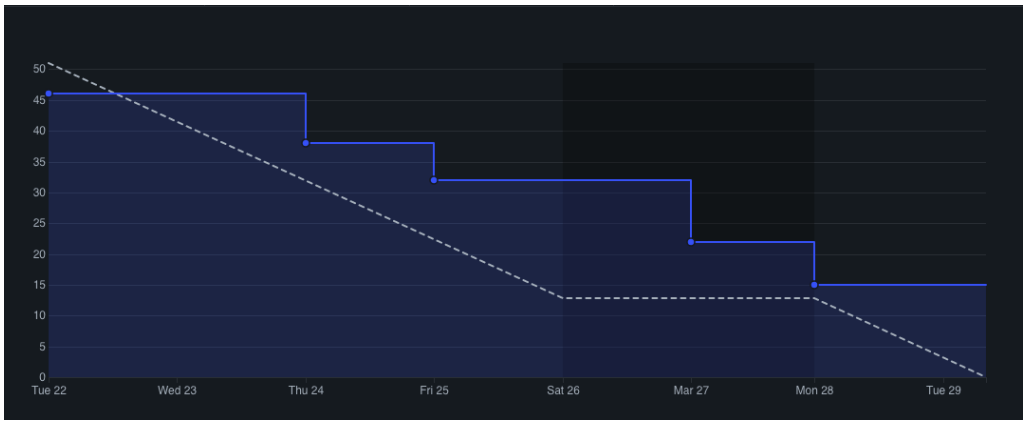


Figura A.16: *Burndown Chart Sprint 15.*

### *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 `UBUMLaaS` y los el aprendizaje semi-supervisado seguro.
  3. Comenzar con la primera etapa de experimentación «seria» que se va a realizar.
  4. Segundo

- **Marcas temporales**

  Este *sprint* se desarrolla entre el veintidós de marzo de dos mil veintidós y el veintinueve de marzo de dos mil veintidós.

- ***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.

### *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 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 veintinueve de marzo de dos mil veintidós y el ocho de abril de dos mil veintidós.

- ***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.

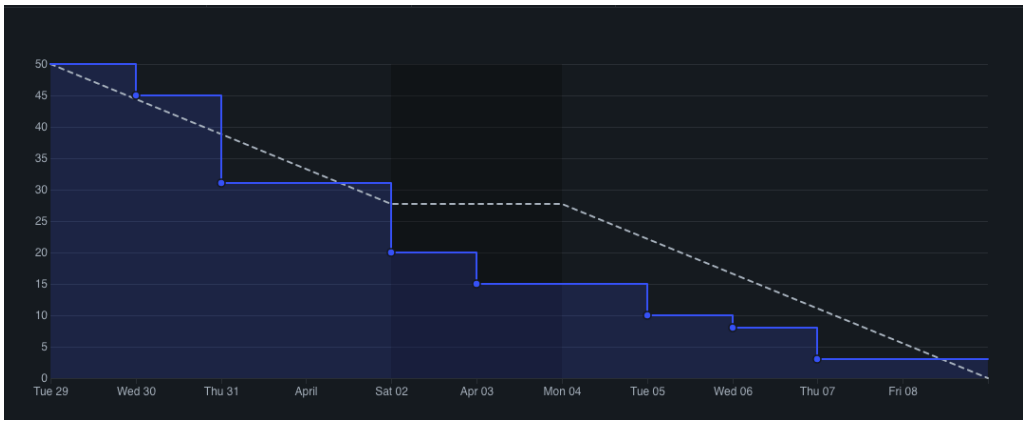


Figura A.17: *Burndown Chart Sprint 16.*

- ***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 LaTeX.
  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 ocho de abril de dos mil veinte y el ocho de abril de dos mil veintidós. Englobando las vacaciones de Semana Santa.

- ***Burndown chart***

- ***Sprint review meeting***

***Sprint* n: Name**

- ***Planning meeting*** Objetivos del n *sprint*:
  1. Primero
  2. Segundo
- **Marcas temporales**
- ***Burndown chart***
- ***Sprint review meeting***

## A.4. Estudio de viabilidad

## Viabilidad económica

## Viabilidad legal

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.

A día de hoy, 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 [7] ó 12207-2-2020 [8]), en contraposición con el *Agile Manifiesto*, 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* [16].

- **Correcto.** Será correcto si, y sólo si, cada requisito declarado se encuentra en el *software* entregado.
- **Inequívoco.** Será inequívoco si, y solo si, cada requisitos 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.

- **Identificable.** 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 librerías 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*,... 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,... 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 del lenguajes de programación como JavaScript,

o lenguajes de marcas como son HTML, CSS, LaTeX... o librerías 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 de garantizar que los modelos calculados son precisos. Además de en caso de pérdidas de conexión, que no ocurran peérdidas 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 monitorizable 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 operatibilidad 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.

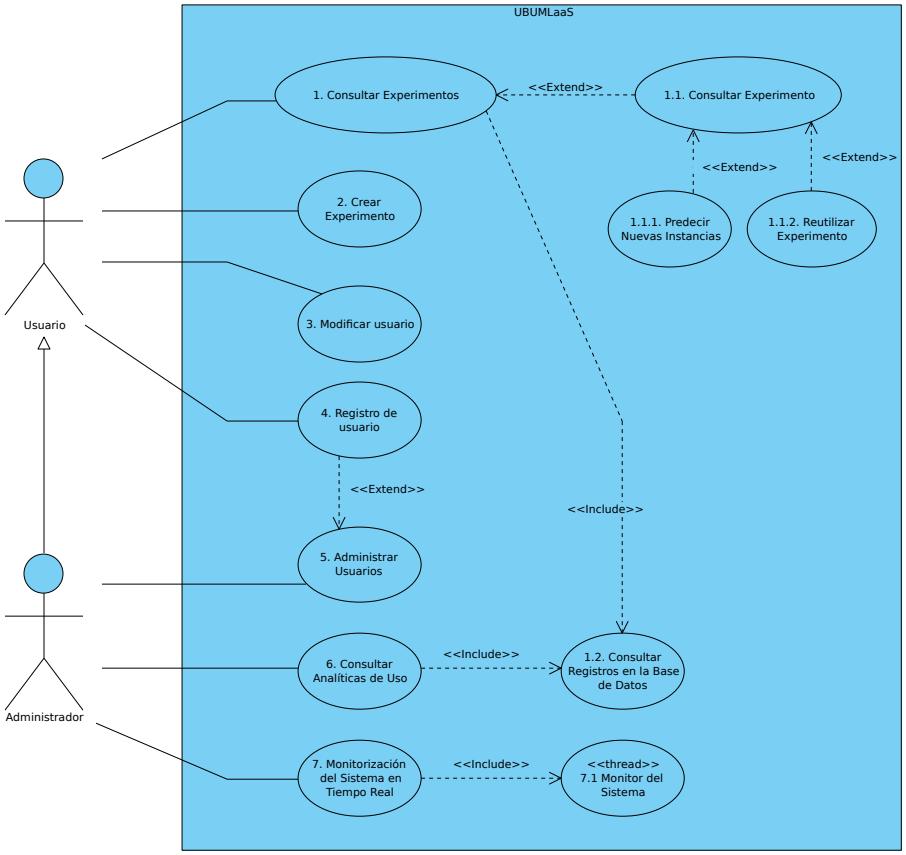


Figura B.1: Diagrama de casos de uso.

### 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** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>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** | ▪ El experimento existe.<br>▪ En caso de haber finalizado y tener métricas de rendimiento, se cargan. |
| **Acciones** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>3. Por cada experimento lanzado se da la opción de ver detalle, reutilizar o eliminar.<br>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** | ▪ El experimento existe y ha finalizado.<br>▪ Las nuevas instancias a predecir tienen los mismos atributos que con las que se entrenó el modelo. |
| **Acciones** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>3. Por cada experimento lanzado se da la opción de ver detalle, reutilizar o eliminar.<br>4. Dentro de ver en detalle hace *click* en «Predict».<br>5. Sube el conjunto de datos a predecir.<br>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** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>3. Busca el experimento del que desea obtener la parametrización para uno nuevo.<br>4. Hace *click* en «Reuse». |
| **Postcondiciones** | 1. El modelo base no se ha visto afectado.<br>2. El usuario se encuentra en la pantalla «Nuevo Experimento» con la configuración «nueva». |
| **Excepciones** | ▪ Algún parámetro interno ha cambiado y ya no se puede reutilizar el experimento.<br>▪ 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** | 1. El usuario entra en la plataforma.<br>2. Realiza alguna acción relacionada con la base de datos.<br>3. Se le devuelven los datos solicitados, y/o<br>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** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Nuevo Experimento».<br>3. Rellena el formulario en función de un conjunto de datos y una técnica de aprendizaje automático.<br>4. Hace *click* en «Crear». |
| **Postcondiciones** | 1. El experimento ha sido añadido a las colas de ejecución.<br>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** | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>3. Hace *click* en «Editar Perfil».<br>4. Modifica los datos como considere oportuno.<br>5. Hace *click* en «Guardar». |
| **Postcondiciones** | 1. Todos los campos son validados de forma que individualmente cumplan sus respectivas restricciones de formato.<br>2. Para aquellos campos que deben ser únicos, se garantiza su unicidad.<br>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** | 1. El administrador entra en la plataforma.<br>2. Hace *click* en «Usuarios» en el panel lateral de administración.<br>3. Hace *click* en «Nuevo Usuario».<br>4. Introduce los datos del nuevo usuario.<br>5. Hace *click* en «Guardar». |
| **Postcondiciones** | 1. Todos los campos son validados de forma que individualmente cumplan sus respectivas restricciones de formato.<br>2. Para aquellos campos que deben ser únicos, se garantiza su unicidad.<br>3. El nuevo usuario recibe un correo electrónico con el *token* 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** | **Administrar 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** | 1. El administrador entra en la plataforma.<br>2. Hace *click* en «Usuarios» en el panel lateral de administración.<br>3. Puede buscar si así lo desea al usuario en cuestión.<br>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 del la plataforma. |
| **Precondición** | Existen estadísticas que mostrar. |
| **Acciones** (para el usuario) | 1. El usuario entra en la plataforma.<br>2. Hace *click* en «Mis Experimentos».<br>3. Hace *click* en «Estadísticas».<br>4. Puede visualizar las estadísticas generales de la plataforma. |
| **Acciones** (para el administrador) | 1. El administrador entra en la plataforma.<br>2. Hace *click* en *«Dashboard»* en el panel lateral de administración.<br>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** | 1. El administrador entra en la plataforma.<br>2. Hace *click* en *«Live Monitor»* en el panel lateral de administración.<br>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

## C.2. Diseño de datos

## C.3. Diseño procedimental

## C.4. Diseño arquitectónico

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-SSL[1]; es por ello que, se dividirá en dos secciones respectivamente, y tantas subsecciones como son necesarias para cada uno de ellos.

## D.2. UBUMLaaS

### 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 las pruebas de integración y despliegue continuo (CI-CD), junto con los ficheros de requisitos para `conda` y `pyenv`.

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

- `/lib/`: librerías utilizadas por el sistema.
- `/lib/is_ssl`: librería propia de métodos de selección de instancias y aprendizaje semi-supervisado.
- `/lib/scikit_ml_learn_data/meka/meka-release-1.9.2/`: librería `Meka` en su versión 1.9.2.
- `/lib/skmultilearn/`: librería `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 librería.
- `/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 pltaforma.
- `/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 seguidos 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 version 3.7, disponible desde [3]. 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 necesitar versiones concretas de determinadas librerías para que todo se integre correctamente con todo y se pueda ver y utilizar como un sistema homogéneo.

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

Tabla D.1: Bibliotecas utilizadas y sus versiones.

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.

### Git

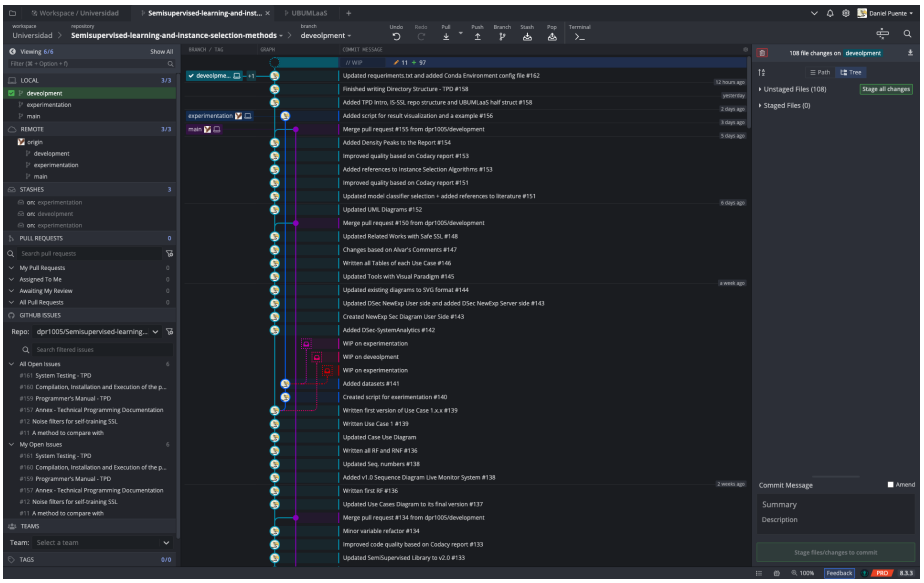

Figura D.1: Interfaz de `GitKraken`.

Para poder utilizar el repositorio ha de utilizarse el gestor de versiones `Git`. Se recomienda utilizar GUI con soporte a VC[3] tales que no requieran de una interfaz de comandos para su utilización, pero eso se deja a decisión del futuro desarrollador.

En la Figura D.1 se aprecia como el *software* `GitKraken` permite, de forma intuitiva y sencilla, el uso de `Git` y todo su potencial. Se puede obtener desde [1].

El repositorio posee una serie diferente de ramas de trabajo, se recomienda para producción utilizar o el último *release* o bien la rama principal.

### Visual Studio Code

`Visual Studio Code` es una herramienta de amplia versatilidad la cual soporta todos los lenguajes de programación utilizados en este proyecto (y

[3]Control de versiones (*Version Control*).

muchos más) de forma nativa; además, posee ciertos *plugins* que facilitan el desarrollo proporcionando *snippets* y similares. Se recomienda su uso ya que es un IDE «todo-en-uno», facilitando las tareas de desarrollo.

Se puede obtener desde [6].

## 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 SSH[4] 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.

- Uso de `GitKraken`.

[4] *Suite* 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.

- 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 dónde 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 [4].
  4. Instalar un cliente SSH compatible con `OpenSSH`. Ver guía [21].
  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 [21].
     *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 correr; 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 entrono 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`, correr el lanzador, y ya está en ejecución.

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

**Nota.** Es importante asegurarnos que todos los ficheros de las librerías 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 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.*

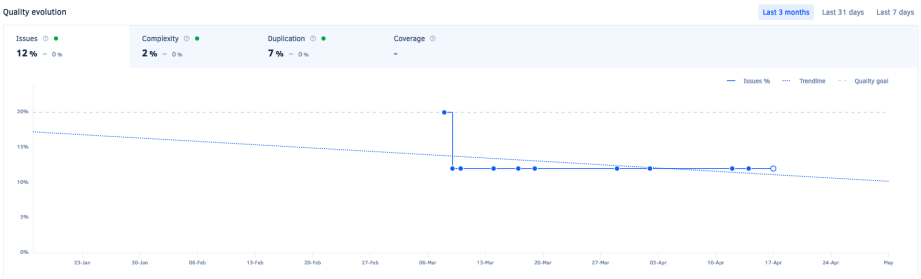


Figura D.2: Codacy.

### 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 CVE[5].

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.*

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

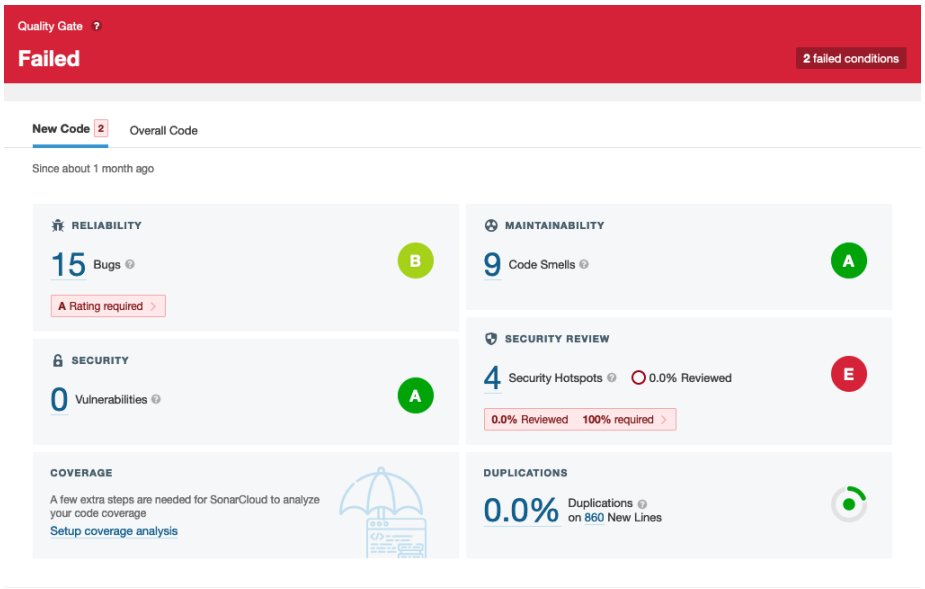


Figura D.3: SonarCloud.

### 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 [5].

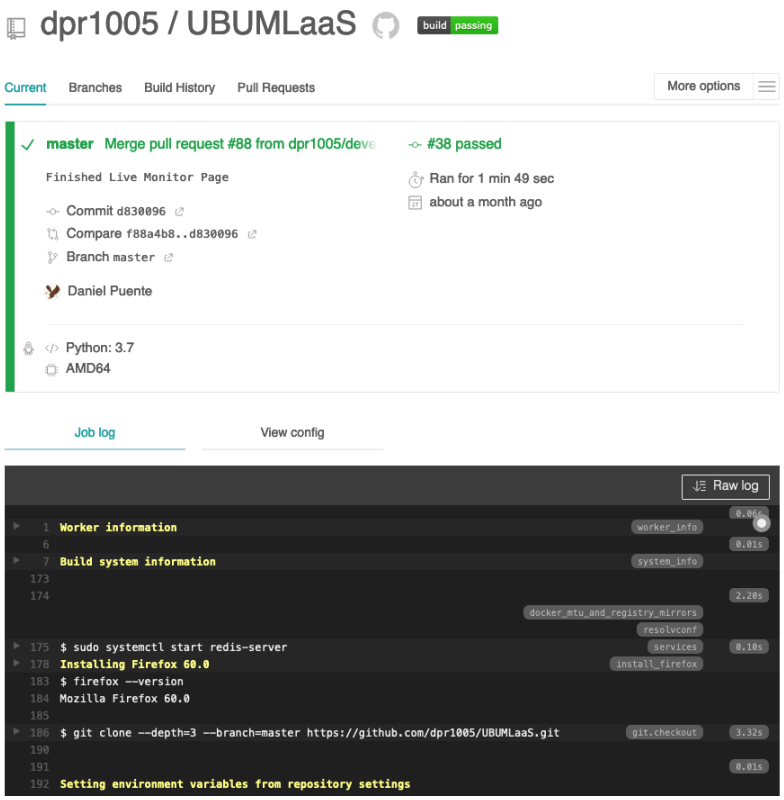


Figura D.4: Travis-CI.

## Pruebas del sistema

Intro

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, debido a que

únicamente comprobaban la interacción del usuario con la plataforma a través de un navegador.

## 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.
- `/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 [3]. 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 librerías de `IS-SSL` se deben utilizar aquellas que se encuentran en negrita.

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.

| Biblioteca | Versión | Descripción |
|---|---|---|
| `matplotlib` | 3.4.3 | Representación gráfica. |
| **`numpy`** | 1.20.3 | Computación de *arrays*. |
| **`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 y sus versiones.

### 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 [6, 2], 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 entrono 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 requeriments.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 D.1, 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 librería 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.

```
from ssl_dnx import TriTraining
from is_dnx 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 D.1: Ejemplo de uso de IS-SSL

## 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 divido 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.

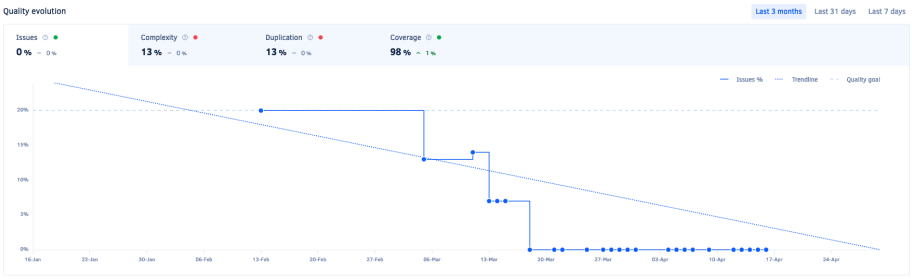


Figura D.5: Codacy.

En la Figura D.5 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 librería sin fallos aumentando su mantenibiliad.

### 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 CVE[6].

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

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

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.

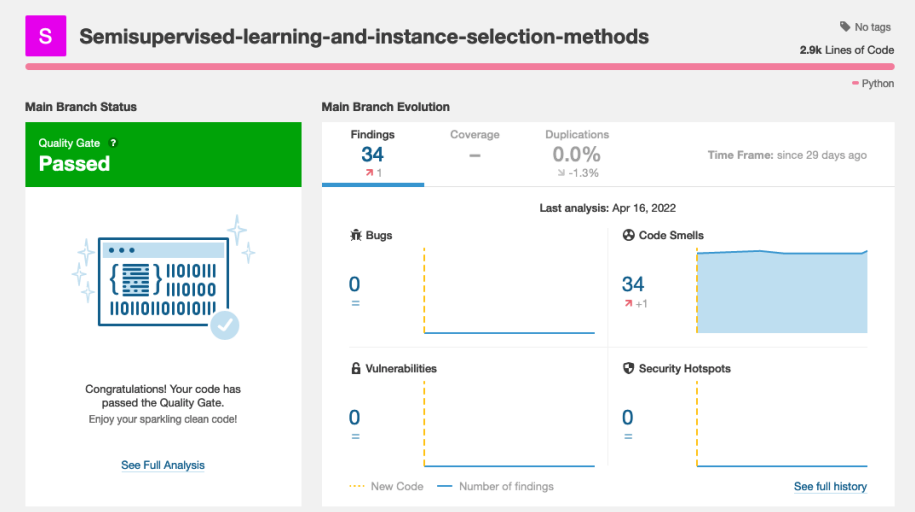


Figura D.6: SonarCloud.

## Pruebas del sistema

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. Requisitos de usuarios

## E.3. Instalación

## E.4. Manual del usuario

# Bibliografía


[1] Gitkraken.

[2] Pycharm.

[3] Python download.

[4] Remote ssh.

[5] Travis-ci ubumlaas.

[6] Visual studio code.

[7] Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, pages 1–40, 1998.

[8] 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.

[9] 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.

[10] 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.

[11] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.

[12] H Frank Cervone. Understanding agile project management methods using scrum. *OCLC Systems & Services: International digital library perspectives*, 2011.

[13] Geoffrey Gates. The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433, 1972.

[14] 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.

[15] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

[16] ISTR Ingeniería Software y Tiempo Real. Ieee830-esp - ctr.unican.es, 2020.

[17] Norbert Jankowski and Marek Grochowski. Comparison of instances seletion algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, pages 598–603. Springer, 2004.

[18] 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.

[19] 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.

[20] Huan Liu and Hiroshi Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115, 2002.

[21] Microsoft. Remote development tips and tricks, 2022.

[22] Dan Radigan. El backlog del producto: la lista de tareas pendientes definitiva, 2021.

[23] Julio Roche. Scrum: roles y responsabilidades, 2020.

[24] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

[25] 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.

[26] 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.

[27] Yan Zhou and Sally Goldman. Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602. IEEE, 2004.

[28] 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.