UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación del Aprendizaje Semisupervisado en el descubrimiento de ataques a Sistemas de Recomendación Documentación Técnica

Presentado por Patricia Hernando Fernández
en Universidad de Burgos — 14 de diciembre de 2022
Tutor: Álvar Arnaiz González

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este documento se pretende mostrar el plan de proyecto seguido a la hora de realizar el trabajo de fin de grado.

Debido al auge de las metodologías ágiles en la industria y sus indiscutibles ventajas, se ha escogido la metodología *scrum* para desarrollar el proyecto. Los conceptos teóricos seguidos se encuentran en el «Manual de *scrum*» [5], temario necesario para obtener la certificación de *Scrum Manager*.

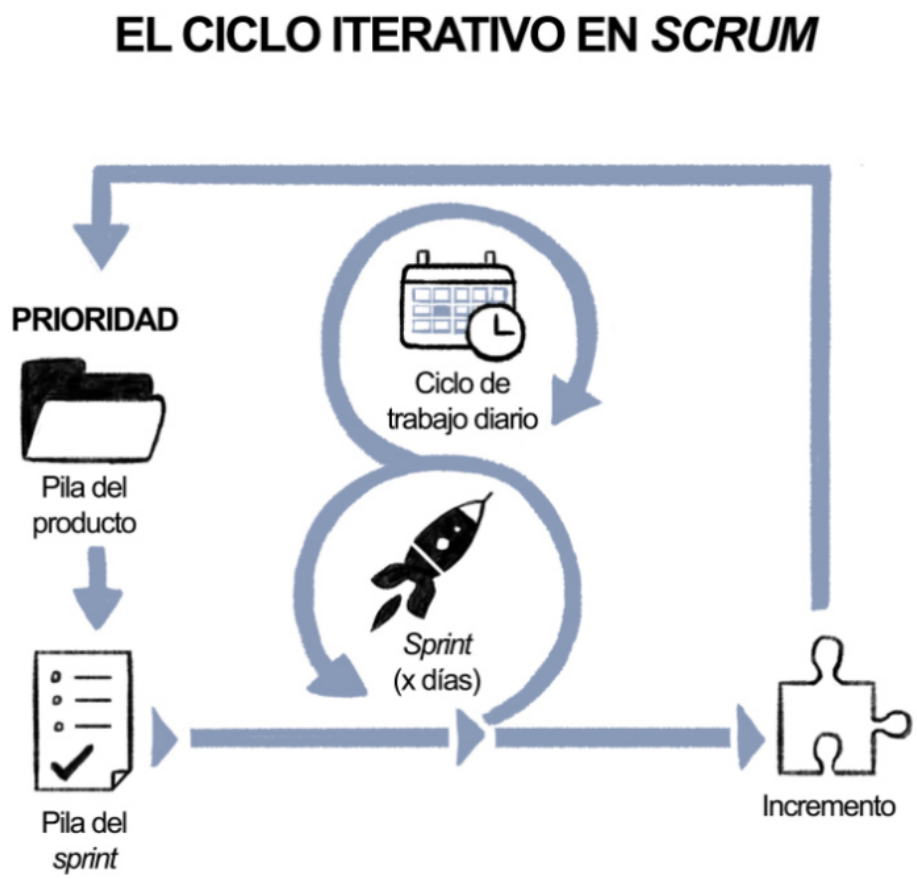
Posteriormente, se expondrá la viabilidad legal y económica del producto.

A.2. *Scrum*

Scrum es una metodología ágil basada en cuatro valores [5]. Sintetizando sus principios, se puede deducir que se pretende valorar a los individuos por encima de las herramientas, el *software* apropiado a la documentación exhaustiva, la colaboración con el cliente y la habilidad de dar respuesta al cambio ante imprevistos.

Siguiendo este esquema, se desarrolla el «ciclo de *scrum*» representado en la imagen A.1, que se puede dividir en varios pilares.

Figura A.1: Resumen del ciclo de *scrum* según el Manual de *scrum* [5].



Roles

Descripción de los posibles interventores durante el desarrollo de un producto.

- *Product owner*: es el representante del cliente, y su responsabilidad es el valor del producto. Es quien gestiona la pila del producto (conjunto de historias de usuario solicitadas por el cliente), así como la prioridad de cada ítem que lo compone.

- *Scrum Master:* es el responsable de garantizar que el proyecto se desarrolla siguiendo los principios de *scrum*, asesorando a los desarrolladores. También es el moderador en las reuniones diarias de *scrum* y el encargado de resolver dinámicas que puedan perjudicar al equipo.

- *Equipo de desarrollo:* es el conjunto de programadores autogestionados encargados de generar los incrementos. Profesionales multifuncionales, deben completar las tareas que se les asignen en el plazo estimado, además de participar en la toma de decisiones.

Artefactos

Son las «herramientas» [5] elementales. Entre ellos se encuentran:

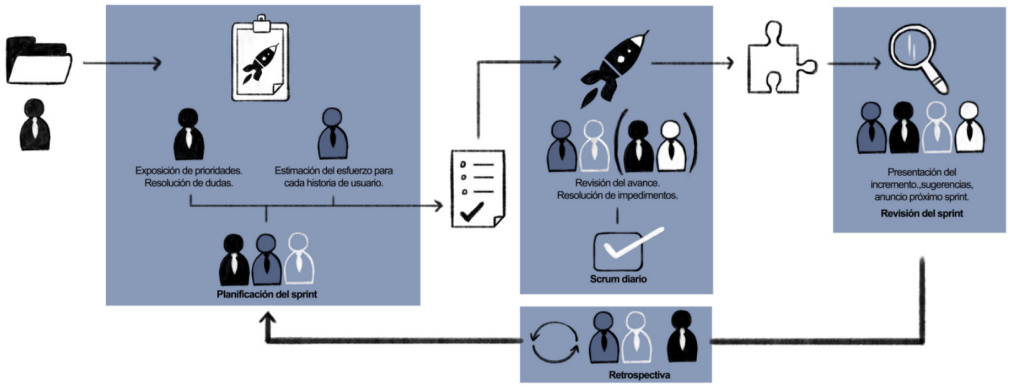
- Pila de producto o *product backlog*: contiene las historias de usuario (el equivalente a los «requisitos» en las metodologías tradicionales). Evoluciona durante el proyecto en función de las peticiones del cliente.
- Pila del *sprint* o *sprint backlog*: es la lista de tareas que han de completar los desarrolladores en un *sprint*. En este proyecto, se han utilizado las *milestones* de GitHub (asignándoles *issues*) para representarla.
- Incremento: resultado de cada *sprint*. Ha de ser un entregable.
- Gráfico de avance o *burn down report*: muestra el trabajo pendiente por realizar en un *sprint* y es actualizado por los desarrolladores. Indica, además, el ritmo de trabajo «ideal» que se debería seguir para alcanzar los objetivos. En este caso, se ha utilizado el gráfico generado por ZenHub.
- Gráfico de producto o *burn up report*: mide cuánto se ha completado respecto al total.

Eventos

Durante el ciclo de *scrum*, se pueden identificar varias actividades que constituyen la rutina y se represetan en la imagen A.2 facilitada por el Manual de *scrum* [5].

1. *Sprint* o iteración: es un periodo de tiempo fijo (y breve) en el que se trabaja para completar una cantidad de tareas prefijadas con antelación (la pila del *sprint*). Se han utilizado los *sprints* de ZenHub para realizar el seguimiento.
2. Reunión de planificación sel *sprint*: marca el inicio de cada *sprint* y determina las tareas a desarrollar, además de su duración temporal.

Figura A.2: Eventos de *scrum* según el Manual de *scrum* [5].



3. *Scrum* diario: breve reunión en la que cada integrante del equipo notifica su ritmo de trabajo con el fin de corregir posibles impedimentos que ralenticen el ciclo. Además, se actualiza el *burndown report*.

4. Revisión del *sprint*: se analiza el incremento entregado y se adapta la pila del producto en caso de necesitarlo (por ejemplo, si se ha encontrado algún *bug* o se necesita refactorizar).

5. Retrospectiva del *sprint*: se aporta el *feedback* necesario para mejorar la siguiente iteración.

Medición

Como se puede observar, cada equipo puede realizar una cantidad de trabajo, generalmente fija, en cada *sprint*. Por ello, es necesario estimar el tiempo que requiere cada tarea y no asignar más trabajo del que se pueda asimilar en cada iteración.

En este proyecto, se han utilizado los «puntos de historia» como unidad de medida.

A.3. Planificación temporal

Planificación por *sprints*

Siguiendo los eventos de *scrum*, y adaptándolos teniendo en cuenta el tamaño del equipo (un desarrollador), se ha decidido planificar el proyecto mediante *sprints*.

Sprint 1: Mustard

- ***Planning meeting***

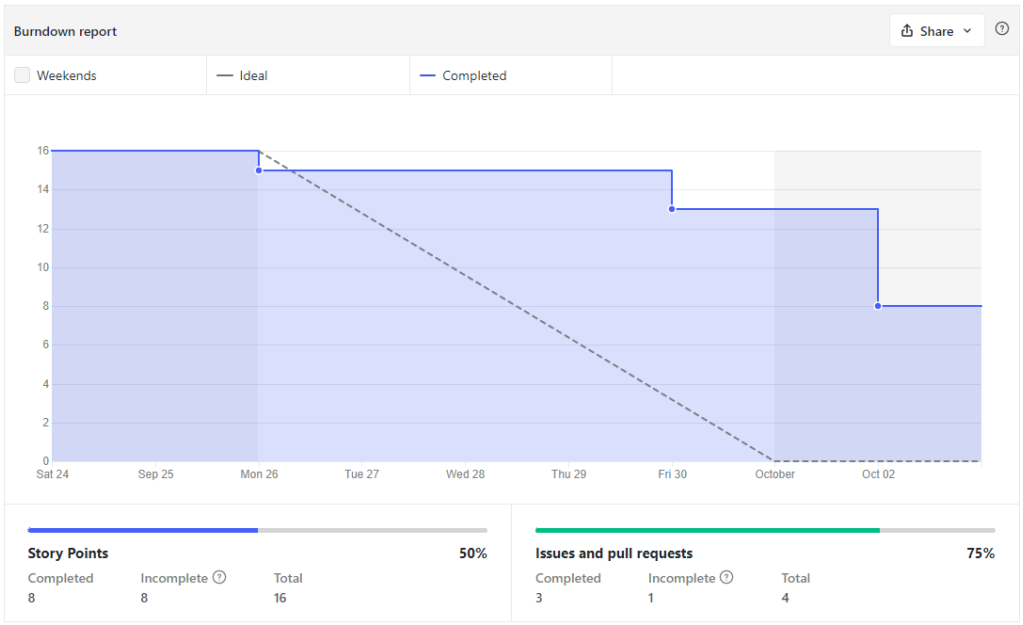
  Durante la reunión se marcaron los siguientes objetivos:

  1. Configuración básica: incluyendo la creación del repositorio, la correcta instalación de ZenHub, la creación de entornos virtuales (miniconda, SKLearn, etc.) y la familiarización con conceptos *scrum*: *milestones, sprints, epics*, etc.
  2. Memoria: comienzo de la redacción incluyendo las secciones de introducción, conceptos teóricos (aprendizaje automático) y trabajo relacionado.
  3. Investigación: búsqueda del código SSADR-CoF y de las bases de datos utilizadas en el paper.
  4. Lectura de papers: Engelen & Hoos [8], García, Triguero & Herrera [6], y Zhou & Duan [9].

- **Marcas temporales** El *sprint* se desarrolló entre el 24 de septiembre de 2022 y el 2 de octubre del 2022.

- ***Burndown Report***

Figura A.3: *Burndown Report Sprint 01*



Como se puede comprobar, no todos los objetivos marcados fueron cumplidos: la estimación del tiempo fue demasiado optimista, además de no contar con el tiempo requerido en solucionar problemas técnicos (LaTeX). Se dejó para próximos sprints la lectura del último paper.

- ***Sprint review meeting*** Durante la reunión se fijaron ciertas correcciones en la memoria (mejorar referencias bibliográficas y la sección de «Trabajos relacionados»), además de la necesidad de introducir una sección teórica de ataques a los sistemas de recomendación.

Sprint 2: Paprika

- ***Planning meeting*** Objetivos del siguiente Sprint:

  1. Configuración: debido a la gran cantidad de tiempo invertida en solucionar errores de compilación en LaTeX, se decidió migrar el proyecto a una nueva instalación basada en Debian.
  2. Correcciones: aspectos estilísticos y completar información.
  3. Lectura: Mingdan y Qingshan [4] con el objetivo de introducir una sección teórica de ataques.
  4. Memoria: redacción completa de los modelos de ataque en los aspectos teóricos.

- **Marcas temporales** El *sprint* se desarrolló entre el 3 de octubre de 2022 y el 18 de octubre del 2022.

- ***Burndown Report***

  En este *sprint* sí se cumplió con los objetivos marcados. Sin embargo, la estimación de tiempo tampoco fue la adecuada, requiriendo más de lo previsto.

- ***Sprint review meeting*** Durante la reunión se resolvieron dudas acerca de bibliografía, referencias y trabajo previo. Además, se acordó empezar a programar, definiendo así los *issues* desarrollados en el siguiente *sprint*.

Sprint 3: Fennel Seeds

- ***Planning meeting***

  Durante esta reunión, se decidió empezar a programar el *co-forest*. Para ello, se definieron los siguientes pasos:

Figura A.4: *Burndown Report Sprint 02*

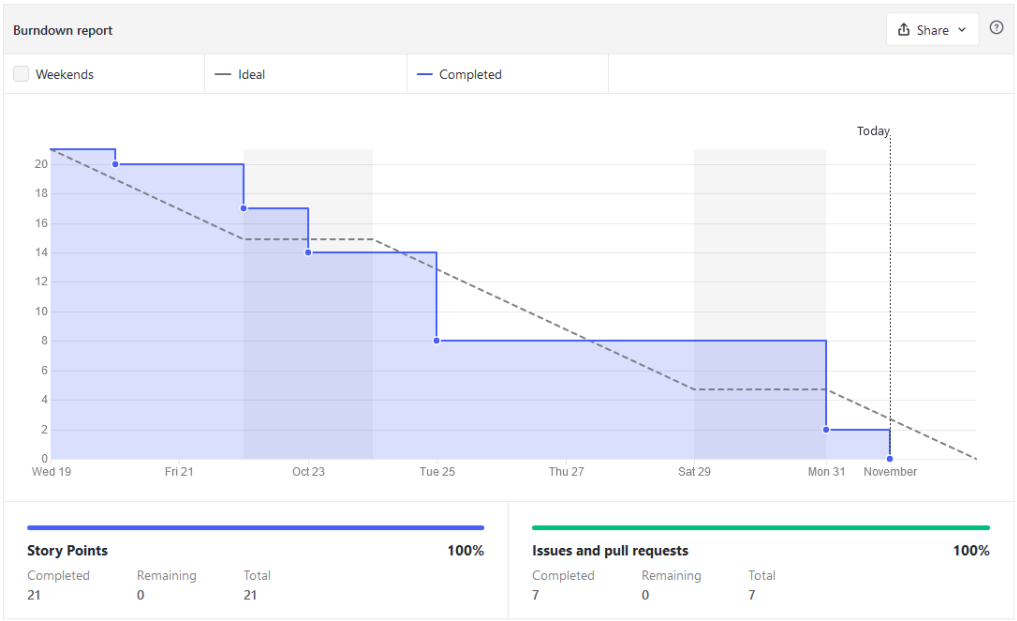
1. Librerías: se acordó aprender a utilizar las librerías más comunes en el *data science*. Entre ellas: MatplotLib y SKLearn. Además, se requirió la correcta configuración del entorno virtual, haciendo que el tiempo dedicado al *issue* fuese mayor de lo estimado (problemas en el `PATH` y con las dependencias).
2. *SKLearn*: aprovechando la correcta documentación de la librería, se decidió repasar los conceptos teóricos básicos, además del manejo de la «interfaz» (métodos comunes). Entre ellos:
   - *Decision trees*
   - *Self training*
   - *Random Forest*
3. Lectura: se concertó la relectura del artículo de Zhou [9] con la intención de comprender el algoritmo y del *paper* «original» del *co-forest* [3]. Durante el proceso de programación, además, se encontró la tesis de Van Engelen [7] y se añadió al conjunto.
4. Documentación: se acordó la corrección de los errores previamente señalados y la inclusión del *sprint* en los anexos.
5. Programación del *Co-Forest*: se programó el pseudocódigo ilustrado en la Tesis de Van Engelen [7], que es muy similar al original [3]

pero con algunas diferencias. Inicialmente se intentó usar el *Random Forest* de *SKLearn*, pero se descartó la idea debido a la poca versatilidad que se ofrecía para manejar los *concomitant ensembles*. Se han de corregir ciertos factores, pero se pospondrá hasta la correcta discusión con el tutor.

- **Marcas temporales** El *sprint* se desarrolló entre el 19 de octubre de 2022 y el 2 de noviembre del 2022.

- ***Burndown Report***

Figura A.5: *Burndown Report Sprint 03*



En este *sprint* se cumplió con los objetivos marcados. Nuevamente, la estimación del tiempo fue inferior a la real (se pensaba que se podría depender más de librerías existentes de lo que se pudo en realidad), calculándose un total de aproximadamente 25 horas reales.

- ***Sprint review meeting***

  Durante la revisión del *sprint*, se llegó a la conclusión de que el pseudocódigo podía ser mejor implementado aprovechando ciertas librerías de *Python*. Se comentó cómo mejorar complejidades espaciales y reducir el código. Se fijaron objetivos para las próximas semanas.

Sprint 4: Cayenne

- ***Planning meeting***

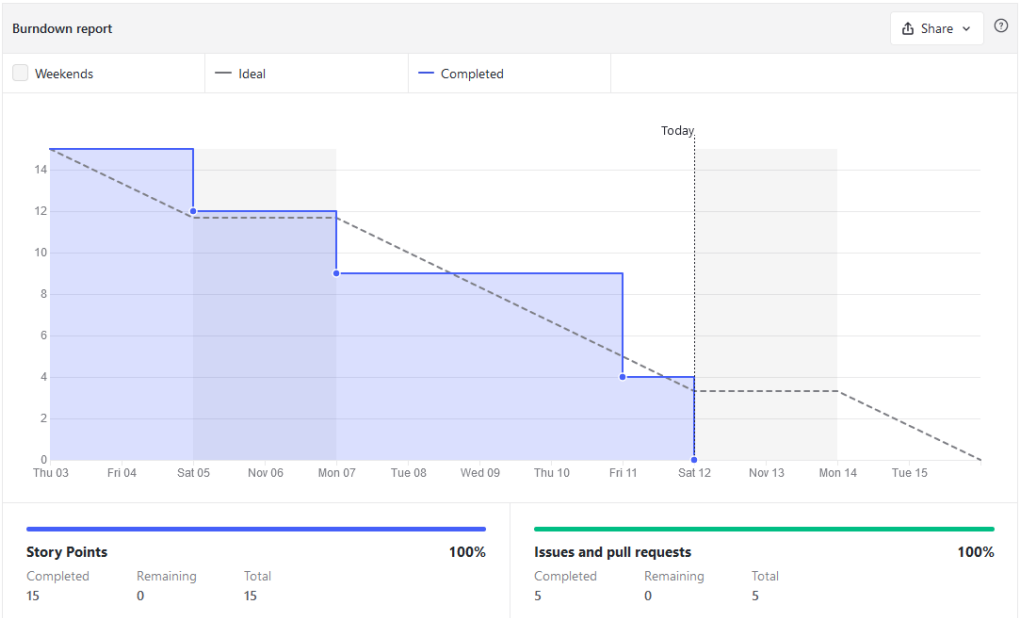
  Durante la reunión se acordaron los siguientes objetivos:

  1. Reimplementación del código: se acordó volver a programar el *co-forest*, esta vez implementando una versión más «*pythoniana*» con el fin de mejorar la complejidad espacial y facilitar la lectura.
  2. Curso de Numpy: se decidió que sería interesante la realización de un curso para aprender a utilizar la librería y aplicarla al código.
  3. Curso de Pandas: aprovechando la relación con el punto anterior, se acordó completar también un curso de esta librería.
  4. Memoria: corregir aspectos anteriores e incluir toda la teoría relacionada con el *co-forest*.

- **Marcas temporales** El *sprint* se desarrolló entre el 3 de noviembre de 2022 y el 15 de noviembre del 2022.

- ***Burndown Report***

Figura A.6: *Burndown Report Sprint 04*



En este *sprint* se completaron los objetivos, aunque quedaron pendientes ciertos aspectos a comentar respecto al código. Destacar que,

debido a que se terminaron antes de lo planeado los *issues* planificados, se aprovechó para modificar ciertos aspectos pendientes relacionados con la memoria y para probar correctamente el código. Esto hizo que el tiempo real dedicado haya sido ligeramente superior al estimado (más tiempo de documentación).

- ***Sprint review meeting*** En la reunión se acordó experimentar con el algoritmo utilizando distintos conjuntos de datos, además de mejorar ciertos detalles de implementación.

Sprint 5: Curry

- ***Planning meeting*** Durante la reunión se acordaron los siguientes objetivos:

  1. Ajustes al código: se acordó mejorar algunos factores, como la generación de objetos «aleatorios» para obtener resultados deterministas en los experimentos, optimización de memoria o corrección de parámetros.
  2. Experimentación: se determinó probar el código con distintos conjuntos de datos en diferentes fases: durante el entrenamiento y tras terminarlo. Para ello, se estudiaron algunos conceptos teóricos y el uso de la librería MatplotLib para representar gráficamente los resultados obtenidos.
  3. Memoria: documentación de los experimentos realizados y correcciones.

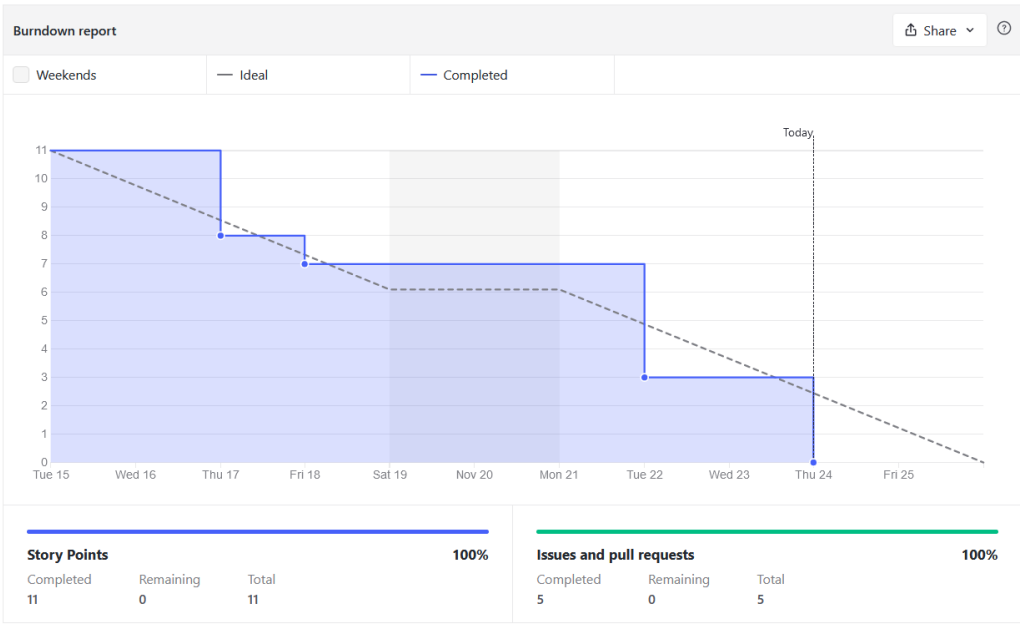
- **Marcas temporales** El *sprint* se desarrolló entre el 15 de noviembre de 2022 y el 25 de noviembre del 2022.

- ***Burndown Report***

  En este *sprint* se cerraron todos los puntos de historia propuestos. Aunque se estimaron 11 horas de trabajo, el tiempo invertido fue superior, realizando 15. La mayor dedicación se justifica por la existencia de reuniones intermedias y de «defectos» encontrados en el código en el trascurso del *sprint*.

- ***Sprint review meeting*** Durante la reunión se acordó comparar los resultados obtenidos con los de otras herramientas, además de empezar el tratamiento de los conjuntos de datos utilizados en el *paper* [9].

Figura A.7: *Burndown Report Sprint 05*



Sprint 6: Coriander

- ***Planning meeting*** Durante la reunión se fijaron los siguientes objetivos:

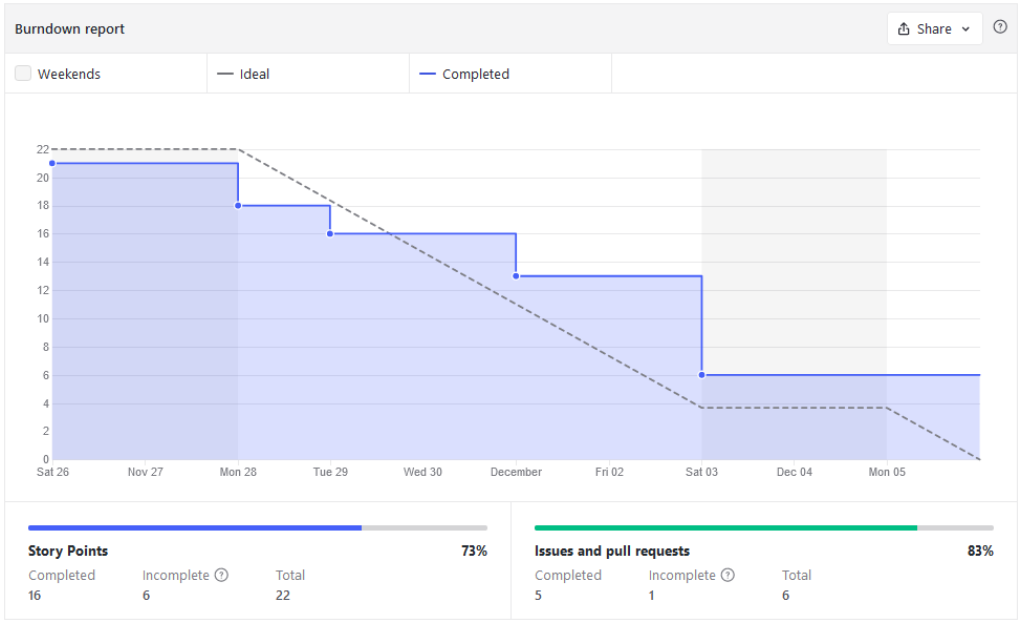
  1. Ajustes en las gráficas: arreglar detalles menores en el formato de ciertos gráficos.
  2. Comparativas: probar los resultados obtenidos y compararlos la herramienta de la Universidad de Granada llamada *Keel*.
  3. *Datasets* «reales»: probar el algoritmo utilizando MovieLens, una de las bases de datos utilizadas en el *paper* [9]. Para ello, es necesaria una re-lectura del artículo y realizar el procesamiento inicial de los datos.
  4. Memoria: documentación de los experimentos realizados.

- **Marcas temporales** El *sprint* se desarrolló entre el 25 de noviembre de 2022 y el 5 de diciembre del 2022.

- ***Burndown Report***

  Puede parecer que los puntos de historia fueron mal estimados debido a que el ritmo de trabajo es bajo. Sin embargo, se justifica debido a

Figura A.8: *Burndown Report Sprint 06*



que durante la experimentación (en concreto, durante la comparativa contra KEEL) se encontraron dos *bugs* en el código (causados no por la lógica del programa, sino por el operador *in* de Python y por no estar preparado para recibir etiquetas que no comiencen en 0).

Debido a que los errores se encontraron una vez se realizó toda la documentación, se tuvo que repetir la sección asociada a los experimentos del *co-forest*, además de localizar y depurar los errores de código. Todo este trabajo supuso un esfuerzo extra de 7 horas, que fueron introducidas en el *backlog* del *sprint* a mediados del mismo (debido a la gran importancia que tienen y la influencia en pasos posteriores). Por este motivo, no se completaron los objetivos previstos. Sin embargo, el tiempo dedicado al proyecto fue el estimado.

Es destacable también que de los 6 puntos de historia que quedan se realizaron 2, pero se decidió dejar el *issue* abierto para el siguiente *sprint*.

- ***Sprint review meeting***

  Habiendo finalizado el modelo, se decidió empezar a experimentar con bases de datos de sistemas de recomendación. Además, se acordó añadir nuevas gráficas.

Sprint 7: Cinnamon

- ***Planning meeting***

  Se marcaron los siguientes objetivos para el *sprint*.

  1. Extraer vectores de características: estudiar e implementar el método de extracción de vectores por ventanas expuesto en el *paper* de Zhou y Duan [9]. Generar ficheros `.csv` con dichos vectores para poder importarlos posteriormente con Pandas. Probar la correcta generación de vectores con perfiles verdaderos y atacantes.
  2. Generación de reseñas de atacantes: siguiendo los modelos estadísticos, generar reseñas de ataque para el *random attack*, el *average attack* y el *bandwagon attack*.
  3. Documentación: añadir introducción y descripción de *scrum* en los anexos. Corregir las sugerencias anteriores. Incluir la descripción del método de extracción de vectores de características por ventanas y la generación de reseñas de atacantes.

- **Marcas temporales** El *sprint* se planificó para desarrollarse entre el 8 y el 20 de diciembre de 2022. Sin embargo y debido al ritmo de trabajo, se cerró el 15 de diciembre de 2022.

- ***Burndown Report***

  Como se puede comprobar, se finalizaron las tareas del *sprint* antes de lo previsto. En gran parte, debido a que el *issue* de extracción de vectores de características estaba ya comenzado en el *sprint* anterior, y requirió menos tiempo del planificado. Además, el resto de tareas no dieron problemas en esta iteración. Debido a que *scrum* no permite añadir nuevos ítems en la pila del *sprint* durante el desarrollo de este, se decidió cerrar y comenzar uno nuevo.

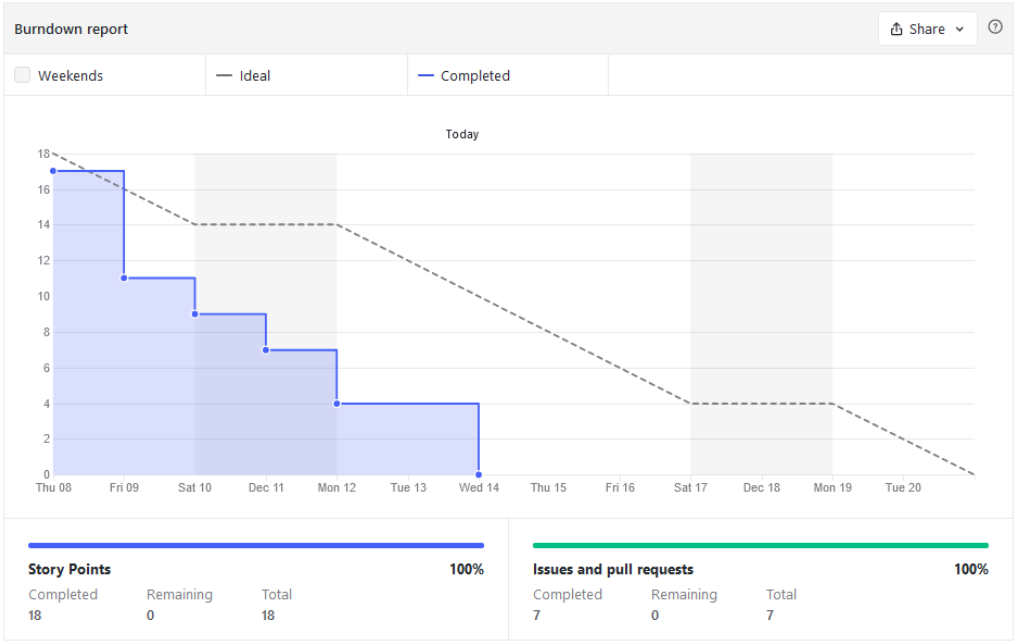
- ***Sprint review meeting***

Sprint N:

- ***Planning meeting***

  1.
  2.

Figura A.9: *Burndown Report Sprint 07*



3.

4.

- **Marcas temporales**
- ***Burndown Report***
- ***Sprint review meeting***

A.4. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

Una muestra de cómo podría ser una tabla de casos de uso:

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

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

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

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Compilación, instalación y ejecución de herramientas auxiliares

KEEL

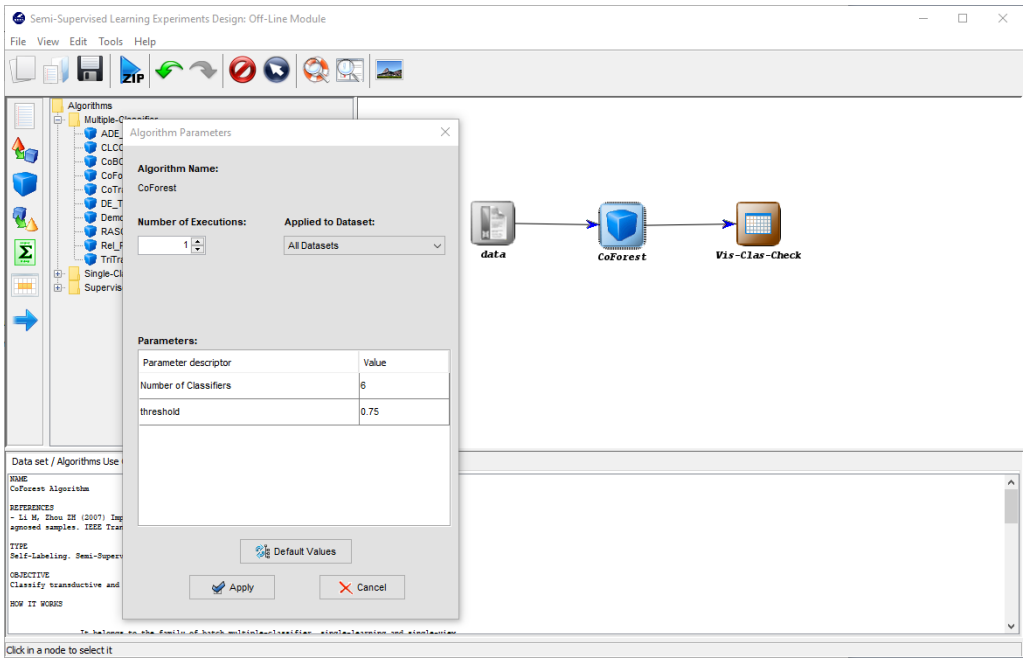
KEEL es una herramienta que permite experimentar con modelos de *machine learning*. Ha sido creada por distintas universidades españolas y financiada por el Ministerio de Educación y Ciencia [2].

Para poder ejecutarla, en primer lugar, se han de descargar los ficheros fuente del repositorio de GitHub [1]. Una vez se han descargado, se compilan aprovechando el fichero `build.xmlz` contenido y la herramienta `ant`.

Mediante el comando `ant cleanAll` se eliminan barios previos (para evitar conflictos), y mediante el comando `ant` se compila el código fuente.

Posteriormente se ejecuta la aplicación mediante el comando `java -jar ./dist/GraphInterKeel.jar` y se utiliza mediante su interfaz gráfica.

Figura D.1: Configuración de un experimento que utiliza el algoritmo *co-forest* mediante la *GUI* de *KEEL*.



D.6. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Jesús Alcalá Fernández (Coordinator). Keel: Github, 2018.

[2] Jesús Alcalá Fernández (Coordinator). Keel: Knowledge extraction based on evolutionary learning, 2018.

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

[4] Si Mingdan and Qingshan Li. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review*, 53, 01 2018.

[5] Marta Palacio. *Scrum Master. Temario troncal 1.* 02 2022.

[6] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowledge and Information Systems*, 42, 02 2015.

[7] Jesper Van Engelen and Holger Hoos. Semi-supervised ensemble learning. master's thesis., 07 2018.

[8] Jesper Van Engelen and Holger Hoos. A survey on semi-supervised learning. *Machine Learning*, 109, 02 2020.

[9] Quanqiang Zhou and Liangliang Duan. Semi-supervised recommendation attack detection based on co-forest. *Comput. Secur.*, 109(C), oct 2021.