Escuela
Politécnica
Superior

Reutilización de datos abiertos de investigación para el desarrollo de sistemas de recomendación

Máster Universitario en Ingeniería Informática

Trabajo Fin de Máster

Autor:
Brayan Stiven Zapata Impatá

Tutor/es:
José Norberto Mazón López
Daniel Ruiz Fernández

Universitat d'Alacant
Universidad de Alicante

Enero 2017

UNIVERSIDAD DE ALICANTE

Máster Universitario en Ingeniería Informática

Trabajo Fin de Máster

Reutilización de datos abiertos de investigación para el desarrollo de sistemas de recomendación

Autor

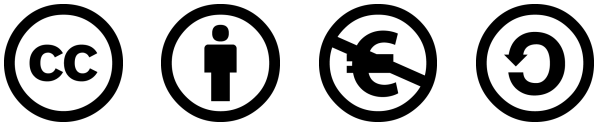
Brayan Stiven Zapata Impatá

Tutores

Dr. José Norberto Mazón López
Departamento de Lenguajes y Sistemas Informáticos (DLSI)

Dr. Daniel Ruiz Fernández
Departamento de Tecnología Informática y Computación (DTIC)

Alicante, Enero, 2017



The goal is to turn data into information,
and information into insight.
Carly Fiorina,
former president of Hewlett-Packard

Agradecimientos

Me gustaría aprovechar la oportunidad de agradecer a las personas que aportaron sus ideas en los inicios de este proyecto: Francisco Javier Cantoral Justo, Antonio Ferrándiz Colmeiro, Virgilio Gilart Iglesias y Daniel Ruiz Fernández. Sin vuestra visión e ideas no habríamos encaminado el proyecto en tan buena dirección. Más especialmente quiero dar gracias a mi tutor principal, José Norberto Mazón López, porque ha sido un apoyo constante lleno de recursos y motivación para continuar adelante. Siempre estaré en deuda con él.

También me gustaría dedicar un momento para agradecer a mi querida *kleintje* por toda la paciencia y el amor que me ha tenido durante estos meses de incesante trabajo. No me olvido de mi fiel compañero *Guinness*, que se ha pasado incontables horas acostado junto a mi silla.

Resumen

Supongamos que somos unos investigadores en informática médica. Como investigadores estamos desarrollando un trabajo con datos clínicos (por ejemplo resultados de pruebas médicas). El objetivo principal de dicho trabajo es el de extraer conocimiento de estos datos. Por ejemplo, sería muy útil disponer de un sistema de recomendación que ayude a detectar ciertas enfermedades como el cáncer. Sin embargo, puede que no tengamos suficientes datos para crear un recomendador que posea ese conocimiento.

En estos casos sería de gran utilidad poder encontrar trabajos donde se hayan empleado datos de una naturaleza similar. Hoy en día se suele intentar resolver esta situación mediante la búsqueda y la lectura de bibliografía especializada. No obstante, a través de este método tradicional de publicación solamente se comparten conclusiones o resultados. En el supuesto que estamos trabajando, lo que necesitamos son datos.

Por otro lado, cada vez es más común encontrar datos abiertos provenientes de resultados de investigación (incluso obligatorio en diversos casos como en proyectos financiados con fondos públicos). La reutilización de estos datos serían de utilidad para nuestro supuesto ya que complementarían su trabajo. Desde hace unos años ya se reconoce que los datos son el núcleo de la economía del futuro y ya es frecuente escuchar que los datos son el nuevo petróleo. Sin embargo, se ha indagado poco en las posibilidades que tiene la reutilización de datos abiertos de investigación y en el valor que se podría proveer con dicha reutilización.

En este proyecto de fin de máster se propone la definición de un proceso de reutilización de datos abiertos de investigación para construir un modelo recomendador con el fin de resolver la problemática anteriormente planteada (además, cabe desta-

car que este trabajo fin de máster se plantea como un punto de inicio en el estudio de la reutilización de datos abiertos de investigación). Para ello se construye una herramienta capaz de explorar un repositorio de datos abiertos del cual seleccionar conjuntos de datos y poder explotarlos. Con este sistema también es posible realizar un *profiling* o exploración de los datos que contenga un conjunto descargado, así como preprocesarlos con técnicas comunes de minería de datos.

Así mismo, la herramienta propuesta es capaz de construir un modelo de aprendizaje automático a partir de los datos que se le suministren y exportarlo de forma que se pueda reaprovechar. De esta manera, el sistema cuenta además con la opción de reutilizar el modelo para lanzar recomendaciones o etiquetar un conjunto de muestras dado, sea privado u obtenido de datos abiertos. Para facilitar la apertura y utilización de la herramienta por la comunidad investigadora, esta se desarrolla con tecnologías multiplataforma y abiertas como son: Python, Tensorflow y Taverna.

Como resultado de este proyecto hemos obtenido una herramienta que cumple con todos los objetivos establecidos y deja lugar a que se continúe trabajando en esta dirección. Además, abre vía a nuevos proyectos como la creación de repositorios de modelos recomendadores en abierto.

Índice general

Capítulo 1

Introducción

En este capítulo se da una breve introducción al trabajo en la sección 1.1. Tras ello, en la sección 1.2 se presenta la motivación del mismo. En la sección 1.3 se verá el estado actual de la materia para continuar en la sección 1.4 definiendo la propuesta del trabajo. Finalmente, en la sección 1.5 se listan los principales objetivos que se intentan conseguir. La sección 1.6 describe la estructura del resto del documento.

1.1. Introducción

En este trabajo final de máster hemos desarrollado un ecosistema de programas con los que un investigador - como podría ser un investigador en informática médica - es capaz de explorar repositorios de datos abiertos, descargar conjuntos de datos, realizar un *profiling* o preprocesamiento de los mismos, así como construir mediante el uso de aprendizaje automático un sistema recomendador para después poder reutilizarlo con otros datos, ya sean abiertos o privados.

Creemos que este sistema puede ser de utilidad a la comunidad científica e investigadora de forma que puedan sacar un mayor provecho del gran potencial de los datos abiertos. Se busca con este proyecto fomentar una cultura de trabajo entre los investigadores de reutilizar y a su vez abrir los datos de investigación. Además de indagar en las posibilidades que ofrece la reutilización de datos abiertos de investigación.

1.2. Motivación

Este documento se presenta como resultado de la investigación y el desarrollo realizado al final del Máster en Ingeniería Informática, con especialidad en *Tecnologías informáticas para la innovación*, cursado entre los años 2015 y 2017 en la *Universidad de Alicante (España)*. La principal motivación de este proyecto viene desde un gran interés personal en los datos abiertos, la inteligencia artificial y su aplicabilidad en otros campos como la salud, además del deseo de trabajar en un proyecto innovador donde aplicar los conocimientos adquiridos en la especialidad cursada.

Supongamos que somos unos investigadores en informática médica. Como investigadores estamos desarrollando un trabajo con datos clínicos (por ejemplo resultados de pruebas médicas). El objetivo principal de dicho trabajo es el de extraer conocimiento de estos datos. Por ejemplo, sería muy útil disponer de un sistema de recomendación que ayude a detectar ciertas enfermedades como el cáncer.

En estos casos sería de gran utilidad poder encontrar trabajos donde se hayan empleado datos de una naturaleza similar. Hoy en día se suele abordar la resolución de esta situación mediante la búsqueda y la lectura de bibliografía especializada. No obstante, a través de este método tradicional de publicación solamente se comparten conclusiones o resultados. En el supuesto que estamos trabajando lo que necesitamos son datos.

Por otro lado existen los datos abiertos que pueden constituir una fuente de información que dirijan una investigación o puedan apoyarla. La reutilización de estos datos serían de utilidad para nuestro supuesto ya que complementarían su trabajo.

Según concluye Fernanda Peset en su presentación sobre datos abiertos en investigación durante el Encuentro de Datos Abiertos en la Universidad de Alicante 2015 [1], es necesario *"desarrollar y utilizar nuevas herramientas de software para automatizar y simplificar la creación y* ***explotación*** *de conjuntos de datos..."*. Este hecho nos alienta a desarrollar una herramienta capaz de encontrar esos datos, explorarlos y utilizarlos en la construcción, en este caso, de un sistema recomendador.

1.3. Estado del arte

Desde hace unos años ya se reconoce que los datos son el núcleo de la economía del futuro y ya es frecuente escuchar que los datos son el nuevo petróleo o el nuevo oro. En reconocimiento de ello, en 2013 los líderes del G8 firmaron lo que se conoce como **La Carta de los Datos Abiertos** [2] para manifestar su deseo de potenciar los datos abiertos.

Destacan desde el G8 en dicha carta el principio de que los datos deben ser abiertos por defecto debido a su alto valor para la sociedad y la economía, comprometiéndose a reorientar sus políticas hacia el cumplimiento de este principio. Además, hacen hincapié en que los datos deben ser publicados en formatos reusables por todos: personas y máquinas, de modo que se pueda estimular la creatividad y la innovación.

Estos deseos coinciden con lo que el comunicado `COM(2014) 442 final 2.7.2014` **"Hacia una economía de los datos próspera"** de la Comisión Europea [3] expresa: vivimos una nueva revolución que está impulsada por los datos digitales. Todas nuestras actividades diarias, al igual que los procesos industriales y la investigación conllevan a la recolección y el procesamiento de datos para crear nuevos productos, servicios y metodologías científicas (como el *Data Driven Research*).

Por ello, desde la Unión Europea se impulsan mandatos y obligaciones a los investigadores que participan en sus proyectos del marco H2020 para que publiquen sus datos de forma abierta, como se indica en la guía de publicación del programa [4]. En apoyo a esta política nace el proyecto **OpenAIRE** [5] que pretende ser un mecanismo para hacerla cumplir. Este portal consiste en una red de repositorios de datos de acceso abierto, archivos y revistas que apoyan también las políticas del acceso abierto a los datos de investigación.

Para cumplir esta función, el CERN colabora con OpenAIRE y desarrolla **Zenodo** [6], un gran repositorio o librería donde los investigadores pueden publicar sus trabajos de forma abierta, admitiendo ficheros en toda clase de formatos y tamaños. Desde el propio sitio se motiva a publicar todo documento o material relacionado con una investigación en favor de la replicabilidad y la credibilidad.

El material publicado en Zenodo se comparte con otro proyecto: **DataCite** [7]. Esta es una organización sin ánimo de lucro que provee de identificadores persitentes (DOIs) a datos de investigación. Su principal objetivo es ayudar a la comunidad científica a localizar, identificar y proporcionar confianza a la hora de citar datos de investigación. Su uso también está apoyado desde la guía de publicación del programa H2020.

Además, DataCite cuenta con un registro de repositorios de datos de investigación llamado **re3data** [8]. Este registro es global y cubre tanto repositorios genéricos como específicos a disciplinas académicas, siendo entonces también un sitio que promueve la cultura de compartir datos en investigación.

Entre otros repositorios de publicación de datos abiertos para investigadores cabe destacar **figshare** [9]. Este es un repositorio que permite a los usuarios subir ficheros en cualquier formato, los cuales identifica con los DOI que provee DataCite, e incluso estos pueden tener previsualizaciones desde el propio explorador. Su meta es que cualquier resultado de una investigación pueda ser divulgado de una forma que el actual sistema académico de publicación no permite.

Una alternativa a Figshare es el repositorio **Dryad** [10]. Dryad es un sitio creado por una organización sin ánimo de lucro que pretende hacer público todos los datos que apoyan las publicaciones científicas de forma que se pueda reutilizar y citar.

De igual manera, la *Open Knowledge Foundation* también cuenta con una plataforma para la publicación de datos abiertos conocida como **Datahub** [11]. Sin embargo, esta web consiste en un repositorio más general y no es específico para la apertura de datos abiertos de investigación, aunque también son admitidos.

Como se puede ver, existe una variedad de repositorios que persiguen unos objetivos más o menos similares: ser fuente donde los investigadores puedan dar acceso abierto a los datos de su investigación de manera que sean reutilizables. Sin embargo, cada uno lo ha desarrollado de una forma diferente por lo que su uso varía. Desde Harvard se desarrolla el proyecto **Dataverse** [12] para proporcionar una herramienta de creación de repositorios de forma que se estandaricen.

Dataverse es una plataforma web de código abierto con la que cualquier organización puede crear un repositorio donde compartir, almacenar, citar, explorar y

analizar datos que puede publicar cualquier autor, recibiendo este todo el crédito a su trabajo. Uno de los objetivos de Dataverse es facilitar también el acceso a los datos a desarrolladores por lo que cuenta con una interfaz API e incluso librerías programacionales en lenguajes como Python, R y Java para trabajar con sus repositorios.

En consonancia con estos trabajos que bogan en dirección a la apertura de los datos en ciencia, se pueden encontrar sitios como **MyExperiment** [13] y **ResearchObject** [14]. Estos consisten en repositorios de experimentos científicos. Ya no solo se publican datos y material relacionado con un proyecto de investigación sino que se admiten esquemas de cómo se han trabajado los datos o cómo se ha conducido la investigación. Con esto apuestan por la replicabilidad de los proyectos.

1.4. Propuesta

La tendencia actual es el desarrollo de sitios web donde poder almacenar y organizar los datos que los investigadores generan o utilizan en sus proyectos. Con ello se pretende que todo aquel que lo desee pueda reutilizarlos. Sin embargo, no hemos encontrado herramientas con las que poder explorar estos repositorios de tal manera que se pueda apoyar el desarrollo de un proyecto de investigación.. La mayoría de herramientas que se están desarrollando se centran en el punto del almacenamiento pero no van más allá (consultar el listado en *Opening Science* para ver más herramientas de este tipo [15]), es decir, no se consideran mecanismos que faciliten la reutilización de datos abiertos de investigación.

Desde el comunicado de la Comisión Europea *"Hacia una economía de los datos próspera"* [3] en la sección 4.2.1, en el punto *2 - Herramientas y métodos de tratamiento de datos*, se dice que:

> *Con el fin de fomentar la I+D en la inteligencia de negocio, los procesos de apoyo a la decisión y los sistemas de ayuda a las pymes y los empredores web, H2020 aborda las analíticas de datos descriptivas y predictivas, la visualización de datos, la inteligencia artificial y las herramientas software y algoritmos de toma de decisiones.*

Así pues, la Comisión Europea reconoce la necesidad de herramientas que no solo apoyen la recolección de datos para que tengan un libre acceso sino que también son necesarias aquellas que permitan trabajarlos para apoyar la toma de decisiones.

Por ello, nuestra propuesta consiste en la definición del proceso de reutilización y el desarrollo de un sistema capaz de explorar un repositorio de datos abiertos del cual seleccionar conjuntos de datos y poder explotarlos. Debido a que este proyecto consiste en un trabajo de fin de máster, se ha acotado el universo de repositorios a explorar en uno solo. El repositorio que nuestro sistema explora es el administrado por Harvard a través del uso de un Dataverse, accesible en `https://dataverse.harvard.edu/`, aunque podría cambiarse por cualquier otro Dataverse con muy poca reconfiguración de la herramienta.

Con este sistema también es posible realizar un *profiling* o exploración de los datos que contenga un conjunto descargado, así como preprocesarlos con técnicas comunes de minería de datos. De esta manera abordamos un punto muy importante nombrado en la cita del comunicado anterior: el sistema realiza analítica descriptiva y permite tener una visualización de los datos.

Así mismo, la herramienta propuesta es capaz de construir un modelo de aprendizaje automático a partir de los datos abiertos que se le suministren y exportarlo de forma que se pueda reaprovechar mediante su aplicación a otros datos de similar naturaleza. De esta manera, el sistema cuenta además con la opción de reutilizar el modelo para lanzar recomendaciones o etiquetar un conjunto de muestras dado, sea privado u obtenido de un repositorio de datos abiertos. Con esta funcionalidad queremos cubrir la necesidad reconocida por la Comisión Europea de crear herramientas software que realicen análisis predictivo y para ello utilizamos algoritmos de inteligencia artificial.

Además, para facilitar la apertura y utilización de la herramienta por la comunidad investigadora, esta se desarrollará con tecnologías multiplataforma y abiertas que permitan su uso por parte de cualquier persona de manera sencilla. Como resultado, tenemos un sistema de apoyo a la toma de decisiones abierto y reutilizable por parte de usuarios de cualquier ámbito, pues solo dependerá de la disciplina de aplicación de los datos usados.

1.5. Objetivos

El objetivo principal del proyecto es el de definir un proceso donde se reutilicen datos abiertos para extraer el conocimiento necesario en la construcción de un sistema recomendador que pueda apoyar un proceso de toma de decisiones. Para conseguir dicho objetivo, se construye una herramienta y se definen unos objetivos secundarios para la misma:

1. Ser capaces de explorar un repositorio de datos abiertos.
2. Proporcionar herramientas para la visualización de datos.
3. Implementar técnicas que preparen los datos para un proyecto de minería.
4. Construir un modelo de aprendizaje automático que sea genérico y pueda aceptar conjuntos de datos abiertos con distintas características.
5. Instrumentalizar el modelo de forma que se pueda reutilizar.
6. Poder realizar recomendaciones con el modelo exportado sobre otros conjuntos de datos.
7. Proporcionar su uso a través de tecnologías multiplataforma y abiertas.
8. Realizar procesados de datos lo más rápidos y eficientes que sea posible.

1.6. Estructura

En este capítulo se ha introducido la materia de este trabajo de fin de máster. Se han presentado trabajos relacionados para describir el estado actual de la misma y así poder dar pie a nuestra propuesta, junto con sus objetivos. En el capítulo 2 veremos las tecnologías utilizadas y la metodología empleada para desarrollar el proyecto. El capítulo 3 describe el sistema a través de un caso de uso. Finalmente, en el capítulo 4 se discuten las conclusiones obtenidas con la elaboración de este trabajo de fin de máster.

Capítulo 2

Tecnologías

En este capítulo se explican las herramientas y técnicas utilizadas en este trabajo, brevemente introducidas en la primera sección 2.1. La sección 2.2 habla sobre Python y las principales librerías utilizadas. En la sección 2.3 se detalla el repositorio utilizado para obtener los datos a trabajar. La sección 2.4 detalla la librería empleada para llevar a cabo el entrenamiento de los modelos de aprendizaje automático. Finalmente, la sección 2.5 trata la herramienta utilizada para englobar el proyecto y facilitar su uso por parte de otros investigadores.

2.1. Introducción

Para el desarrollo de la propuesta se ha elegido el lenguaje de programación Python, motivada dicha elección por la facilidad y rapidez que se puede conseguir en el desarrollo de aplicaciones con este lenguaje gracias a su variedad de librerías.

En cuanto al módulo de exploración del repositorio de Harvard nos hemos ayudado de la propia API que expone Dataverse. Finalmente, se ha decidido utilizar el *framework* Tensorflow de Google para construir el modelo de aprendizaje y el recomendador por la posibilidad de generalizar flujos de trabajo con datos en Python.

Todo esto, además, es construido utilizando Taverna: un sistema de creación de flujos de trabajo científicos. De esta manera, la herramienta propuesta se puede liberar fácilmente a la comunidad y esta puede reutilizarla a través de esta aplicación.

2.2. Python

Figura 2.1: Imagen propiedad de Python Software Foundation [16].

El lenguaje de programación Python es un lenguaje multiplataforma con una licencia de código abierto. Debido a la gran cantidad de módulos y librerías que extienden su funcionalidad, el sistema propuesto se ha desarrollado en este lenguaje en su versión 2.7.6. A continuación se describen las principales extensiones utilizadas para la construcción del proyecto:

- **argparse:** este es un módulo que facilita la creación de interfaces a la hora de pasar argumentos a la aplicación a través de la línea de comandos. Así es posible indicar argumentos opcionales u obligatorios, entre otras opciones como darles valores por defecto.

- **numpy:** paquete fundamental para el cálculo científico que nos proporciona objetos que actúan como vectores multidimensionales, generadores de números aleatorios y otras posibilidades.

- **pandas:** librería que amplía las capacidades de *numpy* con respecto a crear contenedores ofreciendo gran número de funcionalidades para trabajar con conjuntos de datos. Incluye también múltiples herramientas para el análisis de dichos datos.

- **matplotlib:** una de los principales paquetes para la creación de figuras en 2D interactivas y con numerosas opciones de configuración.

2.3. Dataverse

Figura 2.2: Imagen propiedad de Dataverse [17].

Dataverse es una herramienta web *open source* que sirve para crear un repositorio de datos de investigación. Dicho repositorio permite compartir conjuntos de datos, citarlos, explorarlos e incluso analizarlos. La herramienta es desarrollada en Harvard, en el *Institute for Quantitative Social Science (IQSS)*, junto con una comunidad colaboradora.

Para su uso por parte de desarrolladores tiene habilitada una API accesible desde varios puntos [18]. Por un lado tiene un acceso SWORD (*Simple Web-service Offering Repository Deposit*) que permite acceder a Dataverse para subir, consultar o modificar conjuntos de datos. Por otro lado, es posible lanzar consultas directamente sobre el sitio aprovechando su API REST.

En el sistema desarrollado con este trabajo se ha aprovechado la API REST para trabajar directamente con el repositorio puesto que todo lo que se ha necesitado hacer es consultar y descargar datos, siendo ambas funciones posibles por esta vía.

Consulta del repositorio

La consulta del repositorio es muy sencilla. Dado que Dataverse es una herramienta web, tendremos un punto de acceso a la API como puede ser `https://demo.dataverse.org/api/`. Sobre dicha dirección podemos lanzar una consulta añadiendo el término $search?q = breast + cancer$ para obtener un JSON con el listado de conjuntos que estén publicados en el repositorio y coincidan con los términos de búsqueda *breast* y *cancer*.

Es posible indicar otros parámetros en la búsqueda para filtrar, por ejemplo, el tipo de resultado devuelto (colecciones de conjuntos de datos, conjuntos de datos, ficheros) o fechas de publicación.

Consulta de un conjunto de datos

Para ver más información de un conjunto de datos, la API proporciona otro acceso al que podemos enviar una petición indicando el DOI del conjunto. Así, si a la dirección `https://demo.dataverse.org/api/` le añadimos *datasets/* : *persistentId/*?*persistentId* = *DOI* obtenemos un JSON con la información adicional del conjunto que coincida con dicho DOI.

Descarga de un fichero

El último punto de la API aprovechado en este proyecto es el de acceso a ficheros. Al consultar un conjunto de datos específico se obtienen identificadores únicos de sus ficheros. Aprovechando ese identificador, se puede lanzar una consulta como *access/datafile/IDENTIFICADOR* que nos devolverá el contenido del fichero que posea dicho identificador.

Dado que Dataverse acepta multitud de tipos de ficheros en sus conjuntos de datos, existe la posibilidad de que el fichero que se desee descargar no tenga un formato abierto reutilizable. Esto ocurriría con formatos como el PDF, SPSS, Stata... En este proyecto se ha acotado el rango de ficheros posibles a descargar únicamente a los CSV o tipo texto pues son formatos abiertos que se pueden reaprovechar fácilmente con una herramienta automática como la propuesta.

Para poder realizar cualquiera de estas peticiones al Dataverse es necesario indicar en la consulta un token autenticador. Este token es único para cada usuario y lo proporcionará la organización que administre el Dataverse consultado. En el caso propuesto será Harvard al utilizar su sitio Dataverse.

2.4. Tensorflow

Figura 2.3: Imagen propiedad de Google [19].

Tensorflow [20] es una librería de código abierto de cálculo computacional basada en la definición de grafos con flujos de tensores. Inicialmente fue desarrollada por investigadores del *Google Brain Team* para usarla principalmente en proyectos de aprendizaje automático y *deep learning*. Sin embargo, es posible su aplicación en un mayor marco de dominios.

La librería tiene desarrollada una interfaz para Python pero es posible usarla en otros lenguajes como C++ o Go. Una de sus mayores fortalezas reside en su gran portabilidad y la capacidad de aprovechar al máximo el hardware del que dispone la máquina donde se ejecuta. Por ello, es perfectamente posible llevar un código de una máquina donde corre en CPU a otra para que se ejecute en una GPU. Incluso puede trasladarse a plataformas móviles y todo sin tener que modificar el código.

Su uso en este proyecto viene fuertemente influenciado por la capacidad de definir un flujo de datos de forma genérica. Así, el sistema propuesto es capaz de trabajar con conjuntos de datos diversos: distinto número de características, distintos tipos de datos numéricos y diferentes tamaños de muestras. Además, por su portabilidad es ideal para ser usada en distintos sectores científicos donde puedan disponer de diferentes máquinas: pequeños servidores con una sola CPU, máquinas con varias CPUs o incluso con GPUs.

Además, Tensorflow dispone de una herramienta gráfica donde poder visualizar los grafos definidos programacionalmente llamada Tensorboard. Un ejemplo de grafo se presenta en la figura 2.4. Adicionalmente, esta herramienta permite visualizar la evolución de variables como el ratio de error durante el aprendizaje.

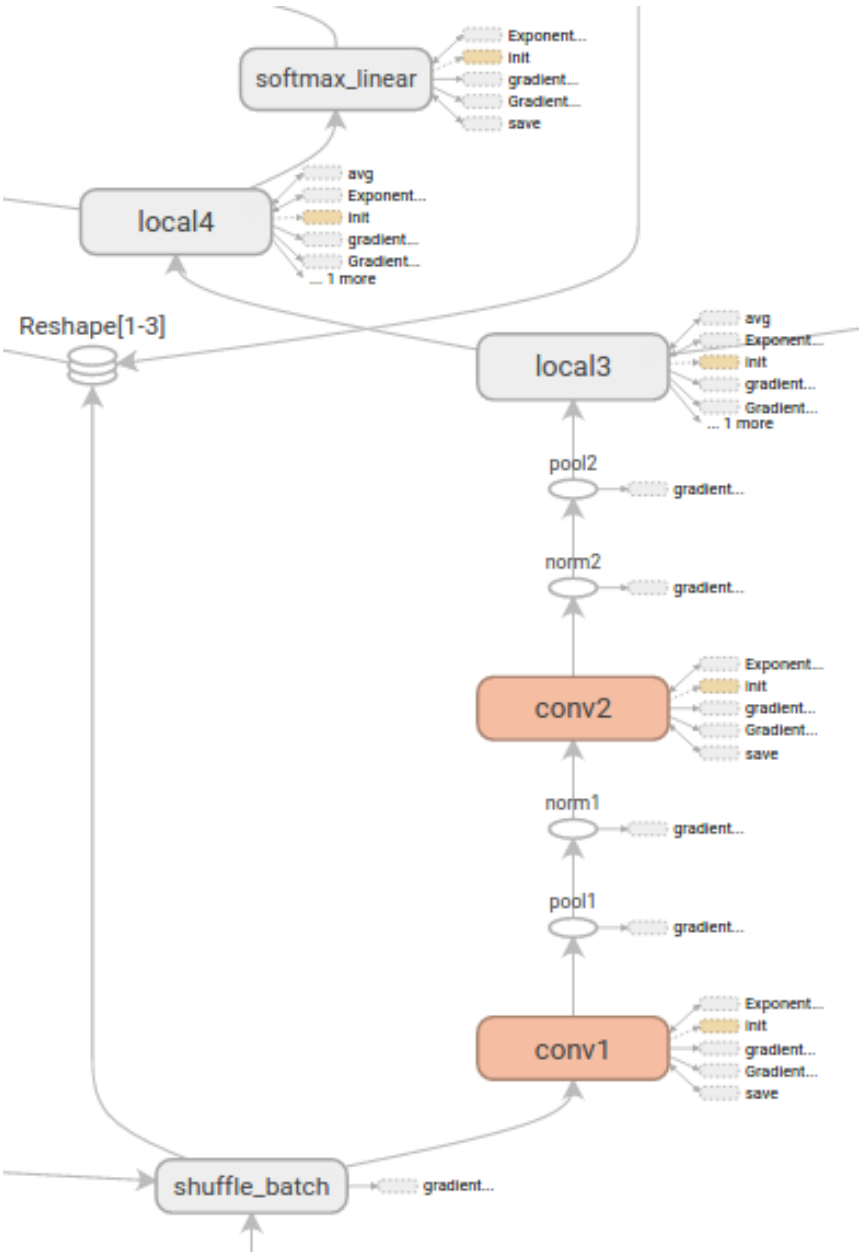


Figura 2.4: Ejemplo de grafo definido en Tensorflow en la visualización de Tensorboard. Imagen propiedad de Google [21].

En el desarrollo del proyecto se ha utilizado la versión 0.11.0, con licencia Apache 2.0, bajo una implementación para Ubuntu 14.04 con CUDA 8.0 y cuDNN 5.1.

2.5. Taverna

Figura 2.5: Imagen propiedad de Apache [22].

Taverna es un sistema de administración de flujos de trabajo o *workflows* que se ejecuta en Java, creado inicialmente por myGrid pero actualmente es ya un proyecto *Apache Incubator*. Este sistema consiste en un conjunto de herramientas con las que un usuario puede crear, buscar y ejecutar flujos de trabajo científicos donde se definen procesos e interacciones entre módulos o incluso el usuario. Se utiliza principalmente en ámbitos como la bioinformática, la química o la astronomía.

El uso de Taverna para desarrollo consiste en la creación de un flujo de trabajo utilizando su aplicación de escritorio. En esta aplicación se pueden incorporar módulos llamados servicios con código en R o Java, llamadas a servicios web usando WSDL o interfaces API REST, manejo de hojas de datos o incluso llamadas de sistema. También incluye algunos servicios conocidos y usados en el ámbito de la biología como BioMart y BioMoby.

El resultado es un diagrama donde se especifican unos datos de entrada y se obtienen una salidas, tal y como se muestra en la figura 2.6. Su uso en esta aplicación nos aporta la posibilidad de definir todos los módulos en Python, usando su servicio llamado *Tool*, y que su interacción con el usuario sea a través de estos flujos. Además, a la hora de liberar la herramienta tan solo es necesario el fichero con la definición del flujo que se genera desde Taverna. Este puede ser depositado en un repositorio como myExperiment, del que hablábamos en la sección 1.3 del estado del arte, pues en este sitio se comparten este tipo de diagramas.

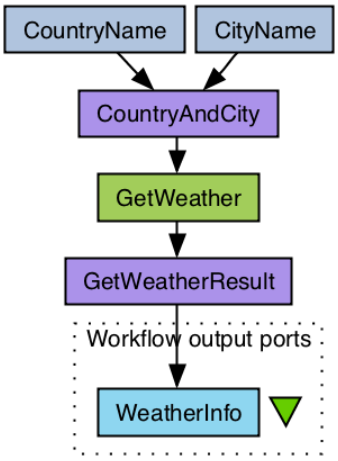


Figura 2.6: Ejemplo de *workflow* científico con Taverna. Imagen extraída de Apache [23].

En este proyecto hemos utilizado Taverna Workbench en su versión 2.5 para sistemas linux Debian, liberado bajo una licencia LGPL 2.1.

Capítulo 3

Propuesta

Este capítulo describe el sistema propuesto al completo a través de un ejemplo de uso. Primero, en la sección 3.1 se introduce el contexto del caso de uso. Entonces, la sección 3.2 detalla la arquitectura de la propuesta y el flujo del proceso. Las secciones 3.3, 3.4, 3.5 y 3.6 describen la implementación de los módulos principales de la herramienta propuesta a través del caso de uso. Finalmente, la sección 3.7 resume cómo puede cualquier investigador reutilizar la herramienta desarrollada.

3.1. Introducción

En el capítulo de introducción hablábamos del caso del investigador en informática médica que disponía de unos datos clínicos y quería trabajarlos para construir un recomendador. En este capítulo, vamos a desarrollar en profundidad este ejemplo de forma práctica con el sistema desarrollado en este trabajo.

Como contexto vamos a tener el supuesto de que tenemos unos datos de los que deseamos extraer conocimiento como para poder clasificar otras muestras. No obstante, no tenemos suficientes datos para construir un sistema recomendador. Estos datos, que podrían venir de una fuente privada o una fuente pública, serán entonces los que llamaremos iniciales y de ellos nos podremos valer para iniciar la búsqueda de más datos. Con ayuda de la herramienta propuesta, procederemos a buscar datos abiertos que nos permitan obtener al final un recomendador.

Para el presente caso, utilizaremos los datos disponibles en el *Machine Learning Repository* de la *University of California, Irvine (UCI)* llamados *Breast Cancer Wisconsin (Diagnostic) Data Set* [24]. De las 699 muestras de este conjunto nos quedaremos con una pequeña parte de 16 muestras por razones que veremos al final del caso de uso.

3.2. Arquitectura de la solución

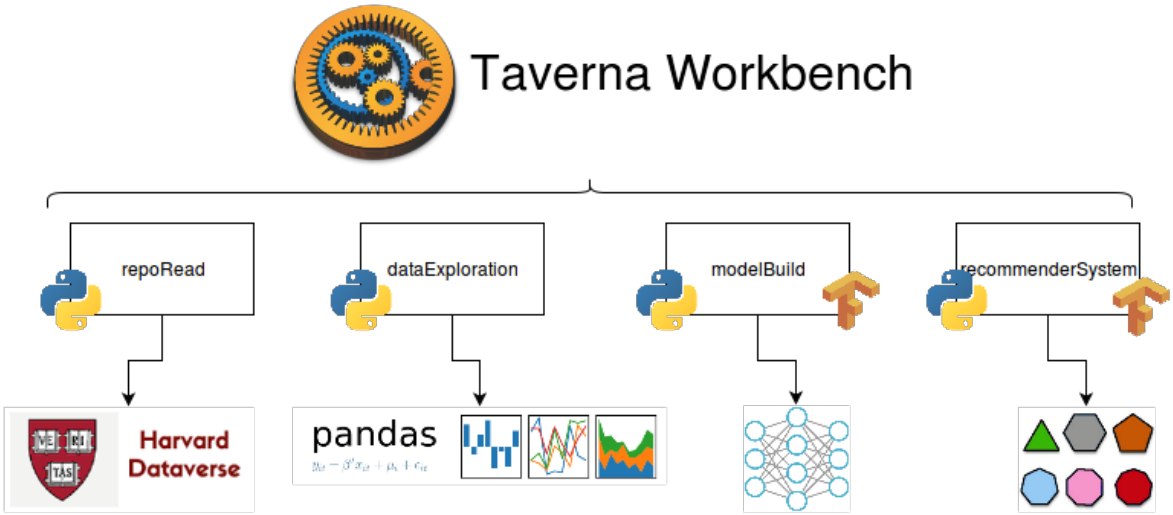


Figura 3.1: Arquitectura del uso de las tecnologías en la herramienta propuesta.

Como se aprecia en la figura 3.1, nuestra propuesta consiste en 4 módulos:

- **repoRead:** funcionalmente este módulo se encarga de explorar el repositorio Dataverse de Harvard. Para ello, tendremos código Python insertado en servicios dentro de un *workflow* de Taverna. Este *workflow* será explicado en más detalle con el caso de uso en la sección 3.3.
- **dataExploration:** en este módulo podremos realizar la visualización de un conjunto de datos además de poder preprocesarlo. Para ello nos valemos de código Python en un nuevo *workflow* de Taverna con el que se puede interactuar. Se describe su funcionamiento en más detalle en la sección 3.4.
- **modelBuild:** con este módulo construimos el modelo recomendador reutilizable dado un conjunto de datos. Esta construcción se realiza con un código escrito en Python que utiliza la librería Tensorflow, todo ello dentro de un *workflow* de Taverna. Esto se explica en más detalle en la sección 3.5.

- **recommenderSystem:** finalmente, gracias a este módulo podemos reutilizar el modelo construido para etiquetar las muestras que deseemos. Está implementado con código Python insertado en un *workflow* de Taverna, que se describe más en profundidad con el caso de uso en la sección 3.6 dentro de este capítulo.

Así pues, el flujo de uso de la herramienta propuesta consiste en la ejecución de *workflows* de Taverna que implementan cada uno de estos módulos. La secuencia natural de uso de estos, que se describirá más en profundidad en las secciones siguientes, es la de explorar el repositorio con *repoRead* para poder descargar un conjunto de datos.

Luego, los datos del conjunto se pueden visualizar y preprocesar con el módulo *dataExploration* para pasar a entrenar el modelo de aprendizaje con *modelBuild*. Este último módulo exportará el recomendador de forma que el módulo *recommenderSystem* pueda apoyarse en él para realizar el etiquetado de las muestras que disponíamos inicialmente.

3.3. Exploración del repositorio

Para realizar la exploración del repositorio Dataverse de Harvard en nuestro sistema disponemos del módulo **repoRead**. En la figura 3.2 mostramos el flujo del proceso en Taverna. Las entradas, llamadas *puertos* en Taverna, de este *workflow* son:

- **keyWords:** estas serán las palabras clave para lanzar la búsqueda sobre el repositorio.
- **headerFile:** es posible indicar un fichero de datos que tenga una cabecera para que se usen esos términos para dar una segunda valoración de la relevancia de un resultado de la búsqueda. En nuestro caso de uso no tenemos cabeceras pero se dará un ejemplo de su uso al final de la sección.

El flujo básicamente consiste en lanzar la consulta sobre el repositorio y nos irá mostrando los resultados uno a uno. Al mostrar un resultado nos dará la opción de saltarlo para ver el siguiente, descargar el archivo de datos adjunto o salir del programa. Para verlo mejor, usaremos este módulo con nuestro caso de uso.

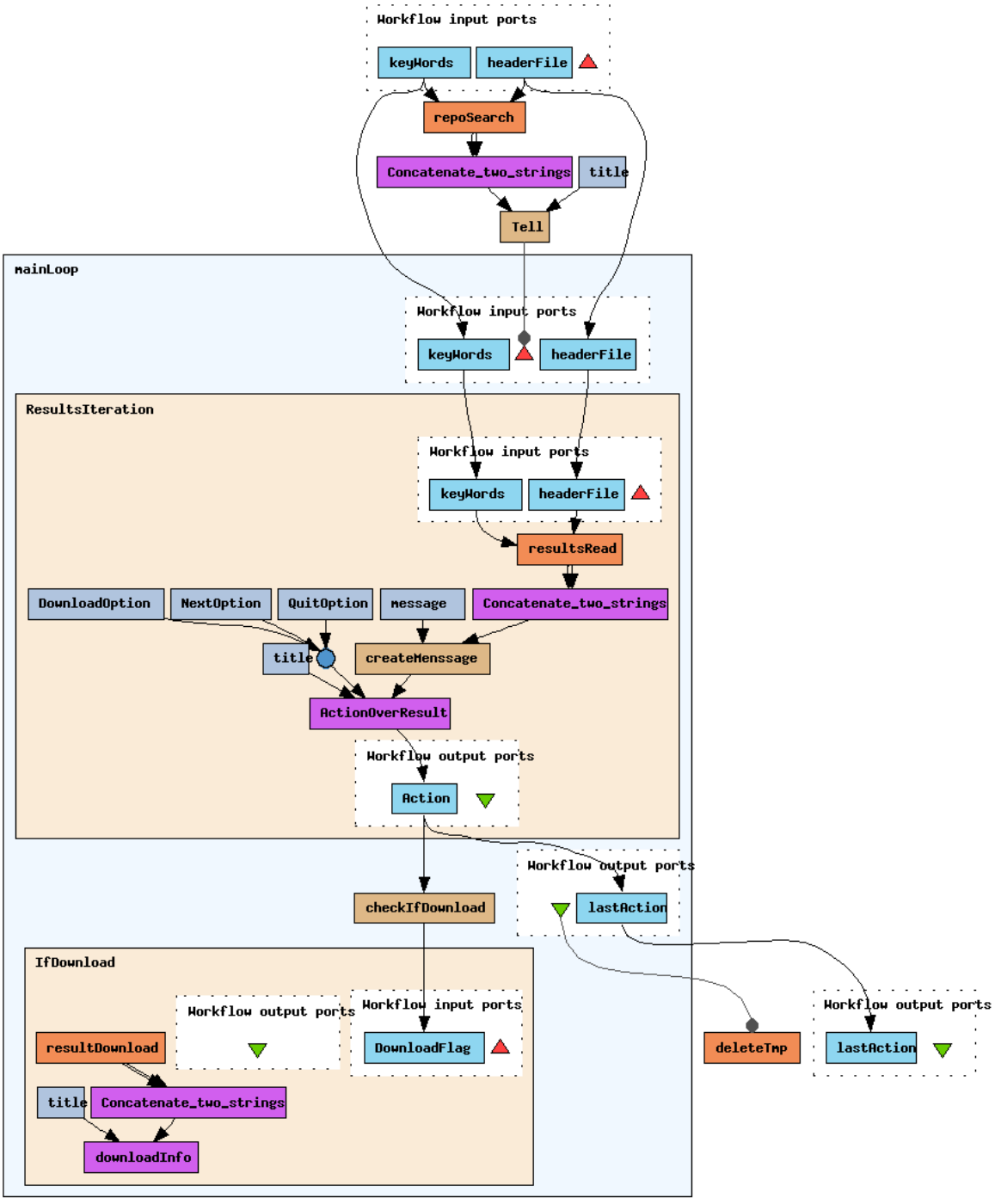


Figura 3.2: Diagrama con el flujo del proceso *repoRead* en Taverna.

Lanzamos el flujo indicando para *headerFile* una cadena vacía ya que nuestros datos iniciales no tienen una cabecera por lo que no hay términos que aprovechar. En cambio, para el puerto *keyWords* indicamos la cadena `'breast cancer'` ya que nuestros datos son resultados clínicos sobre cáncer de mama. En la figura 3.3 se pueden ver los cuadros de Taverna donde se indican estos datos. Con ellos introducidos solo queda pulsar sobre *Run workflow* para que se inicie la ejecución del módulo.

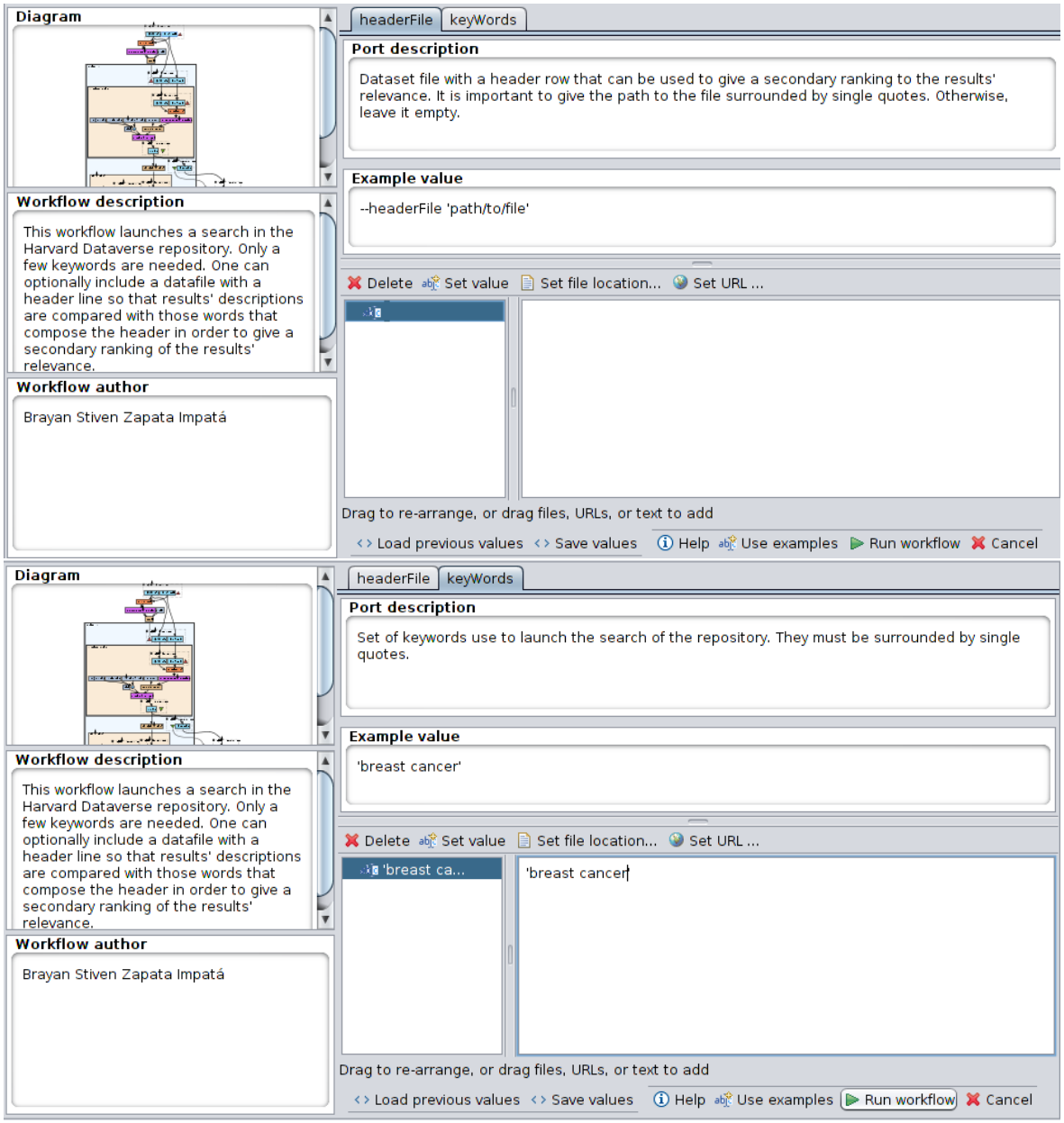


Figura 3.3: Entrada de datos para lanzar la búsqueda sobre el repositorio.

Una vez se inicia la búsqueda, el módulo lanzará un aviso indicando cuántos conjuntos ha devuelto el repositorio, mostrado en la figura 3.4.

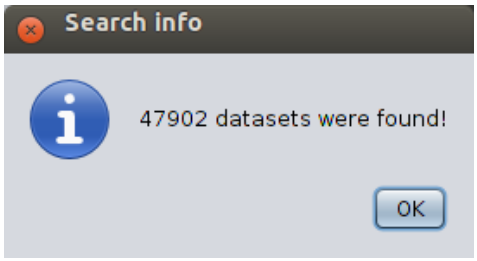


Figura 3.4: Aviso del número de resultados encontrados por la búsqueda.

Al pulsar sobre `Ok` el módulo empezará a mostrarnos los resultados uno a uno en ventanas como la que se ve en la figura 3.5. Esta ventana incluirá como datos del conjunto encontrado:

1. **Search score:** esta es una puntuación que asigna Dataverse a la relevancia que posee el conjunto con respecto a los términos usados para la búsqueda, por lo que los resultados vienen ordenados de mayor a menor importancia para ver antes lo que posiblemente sea más interesante.

2. **Global Id:** este es el identificador global del conjunto.

3. **Description:** si el conjunto se ha publicado con una descripción, esta se muestra.

4. **Matches - Matched words:** si no hemos indicado un *headerFile*, entonces el programa intenta buscar los términos de la búsqueda en la descripción para dar una segunda calificación al resultado.

En este punto lo que podemos hacer con el conjunto son dos acciones sencillas: o bien intentamos descargar el fichero adjunto (debe ser un fichero CSV o de texto para poder ser reutilizado) o vemos el siguiente resultado devuelto por el repositorio. También podemos salir del programa. Para ello, solamente hay que seleccionar la opción deseada del desplegable abajo de la ventana y pulsar `Ok`.

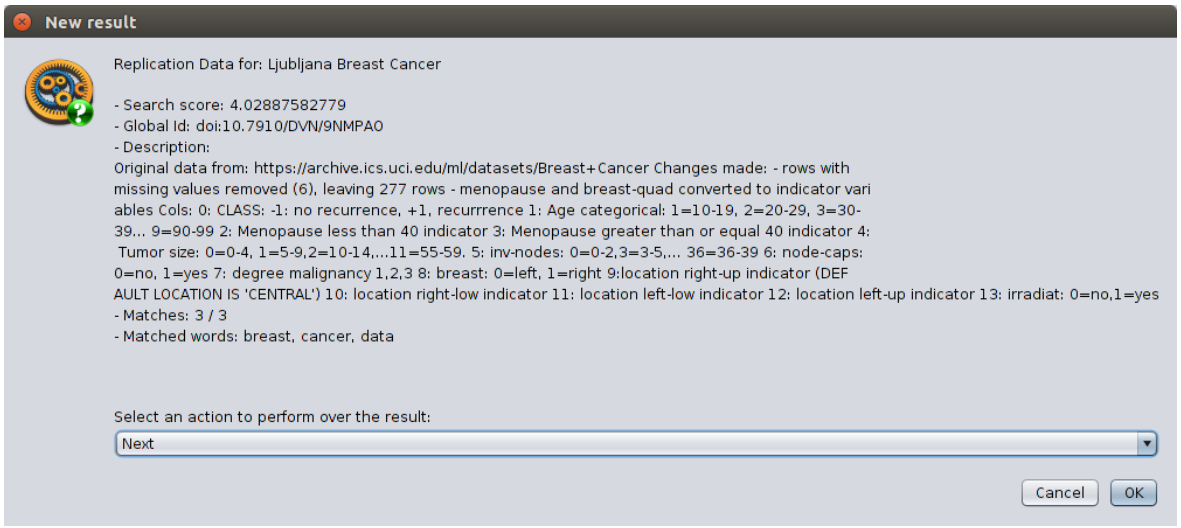


Figura 3.5: Ejemplo de visualización de un resultado devuelto por el repositorio.

Si elegimos mostrar el siguiente resultado podremos acabar encontrando el conjunto de datos descrito en la figura 3.6. Este conjunto consiste en una muestra modificada de nuestro conjunto inicial. Según se aprecia en la descripción, se han eliminado muestras con datos que poseían el valor `'?'`, entre otras modificaciones.

En nuestro caso se podría utilizar este conjunto encontrado en el repositorio para entrenar un modelo y luego lanzar recomendaciones sobre nuestras muestras. Así pues, marcaremos `Download` y al pulsar `Ok` el sistema procederá a descargarnos el fichero adjunto.

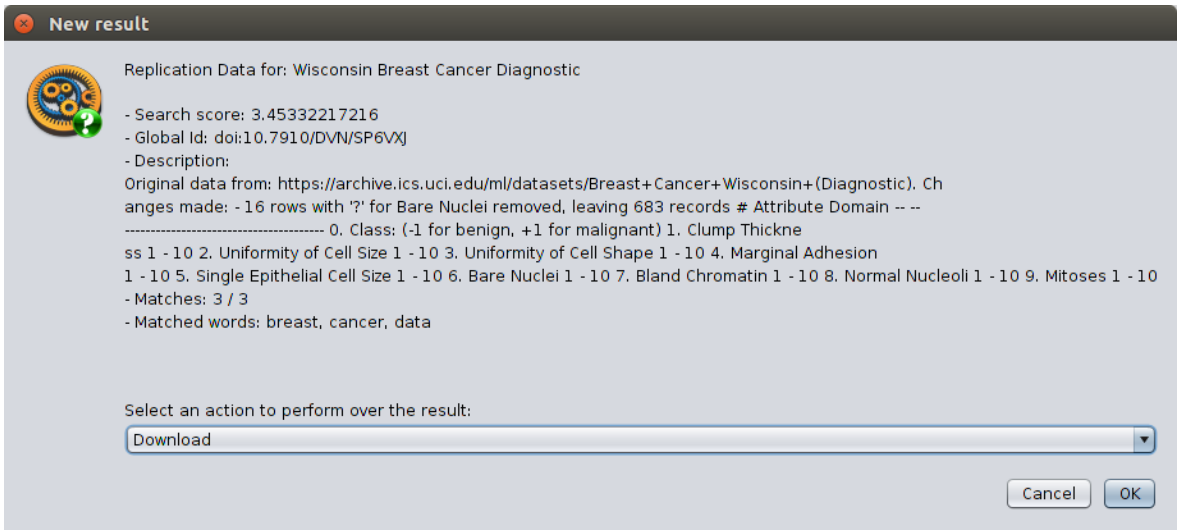


Figura 3.6: Conjunto de datos relacionado a nuestros datos iniciales.

Si el conjunto posee un fichero de datos adjunto en un formato reutilizable (como es CSV) entonces el sistema lo descargará y nos informará del nombre que le ha dado a este, como se ve en la figura 3.7. Este fichero se depositará en un directorio llamado `datasets` en una ruta del equipo configurada dentro del componente **resultDownload** del *workflow* de Taverna.

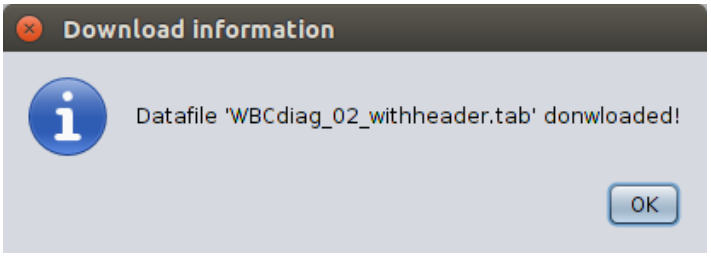


Figura 3.7: Aviso tras la descarga exitosa del fichero de datos de un resultado.

Si por el contrario el conjunto de datos tiene un fichero de datos en un formato no abierto como es PDF, SPSS o Stata, el sistema nos avisará de dicha situación y pasará al siguiente resultado, como se muestra en la figura 3.8.

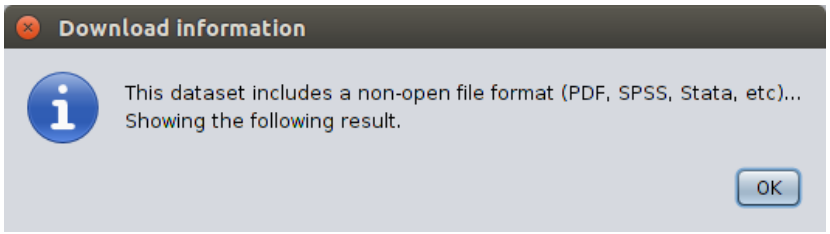


Figura 3.8: Aviso tras la descarga fallida del fichero de datos de un resultado por su formato.

Como comentábamos al inicio, es posible indicar un fichero con una cabecera por el puerto *headerFile*. El fichero recién descargado tiene una cabecera así que podemos utilizarlo para escenificar el uso de este dato de entrada. Para ello, tal y como se ve en la figura 3.9, indicamos el valor `--headerFile 'RUTA/FICHERO/DESCARGADO'` para este puerto.

En cuanto empieza el sistema, aparte de avisarnos del número de resultados, nos informará cuales son los términos encontrados en la cabecera del fichero indicado, como se muestra en la figura 3.10. Con esa información, los datos de *matches / matched words* en los resultados cambiará como se ve en la figura 3.11.

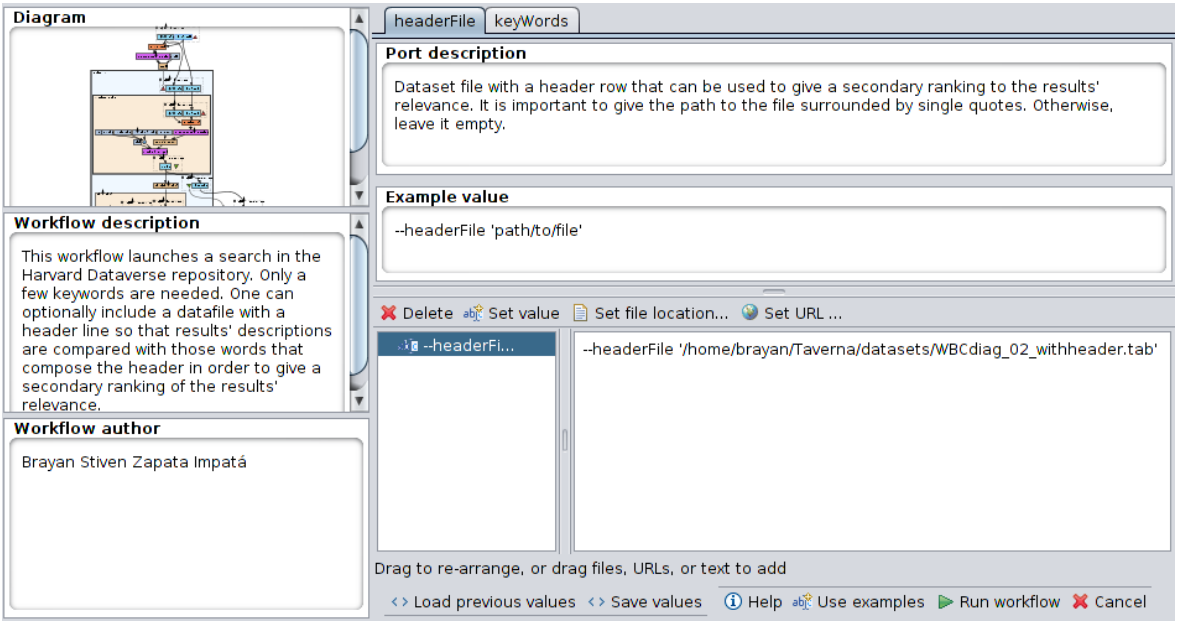


Figura 3.9: Ejemplo de uso del puerto de entrada *headerFile*.

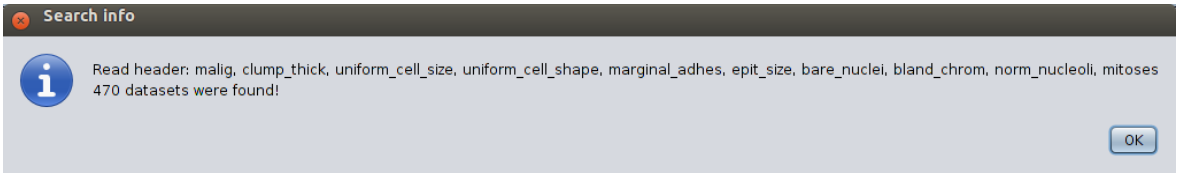


Figura 3.10: Aviso con información de los términos encontrados en la cabecera de *headerFile*.

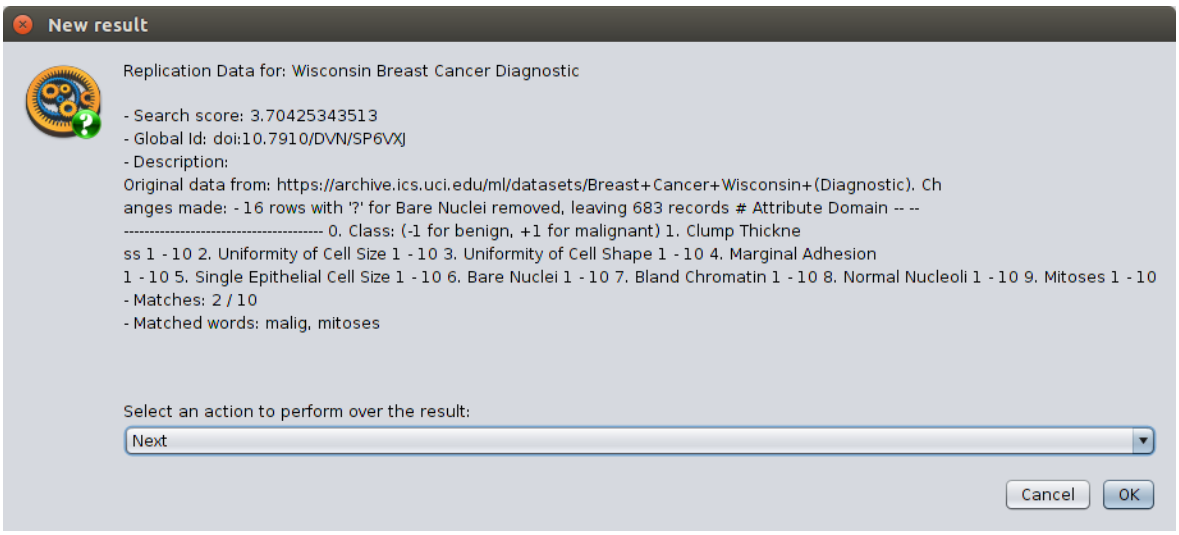


Figura 3.11: Resultado donde se ven las coincidencias entre la descripción y el *headerFile*.

Al final de esta etapa habremos acabado con un fichero nuevo en nuestro entorno de trabajo. Ahora, podemos proceder a realizar una exploración de sus datos y preprocesarlo si es necesario para después continuar con la construcción del modelo recomendador.

3.4. Visualizado y preprocesado de los datos

Para agilizar el uso del fichero descargado, vamos a suponer que lo hemos llamado `replication data wisconsin.tab`. Con este fichero continuaremos con el flujo del proceso de la propuesta para ver ahora el módulo que realiza la visualización y el preprocesado de datos. El diagrama principal se presenta en la figura 3.12.

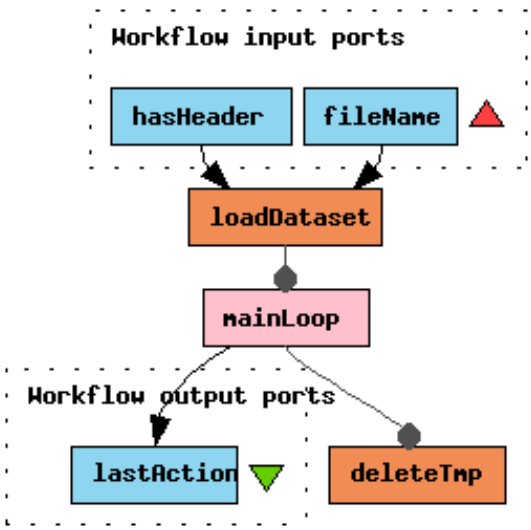


Figura 3.12: Diagrama con el flujo del proceso *dataExploration* en Taverna.

Como se puede ver, este *workflow* requiere solamente de dos entradas:

- **fileName:** ruta absoluta al fichero que contiene el conjunto de datos a tratar.
- **hasHeader:** en caso de que el fichero tenga una línea de cabecera, se ha de indicar en este puerto `--header`, sino se deja vacío y el sistema le asigna una cabecera enumerada desde 0.

Su ejecución consiste en una carga inicial del conjunto para crear unos ficheros auxiliares que agilicen su tratamiento con la cabecera. Tras esto se entra en un bu-

cle central, llamado *mainLoop*, donde se desplegarán todas las opciones del módulo. Cuando se termina con el programa se realiza un paso de limpieza de ficheros auxiliares para finalizar. En la figura 3.13 presentamos el diagrama con el flujo del *workflow* anidado *mainLoop* visto en la figura 3.12.

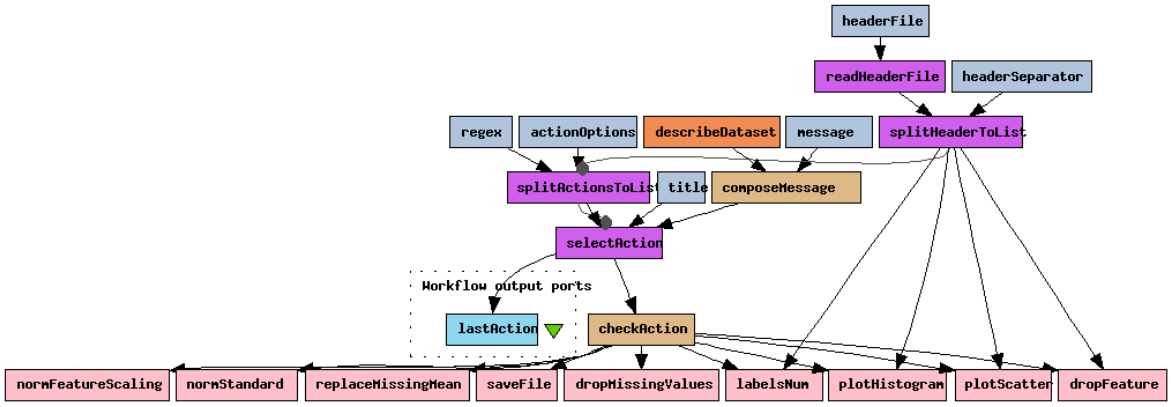


Figura 3.13: Diagrama con el flujo del proceso central que compone *dataExploration*.

Para ver mejor qué funcionalidad tiene este flujo continuaremos con el caso de uso. Con dicho objetivo, realizamos una ejecución del *workflow* en Taverna indicando para *fileName* la ruta al fichero descargado, como se ve en la figura 3.14. Para el puerto *hasHeader* indicamos el valor `--header`, como se ve en la figura 3.15.

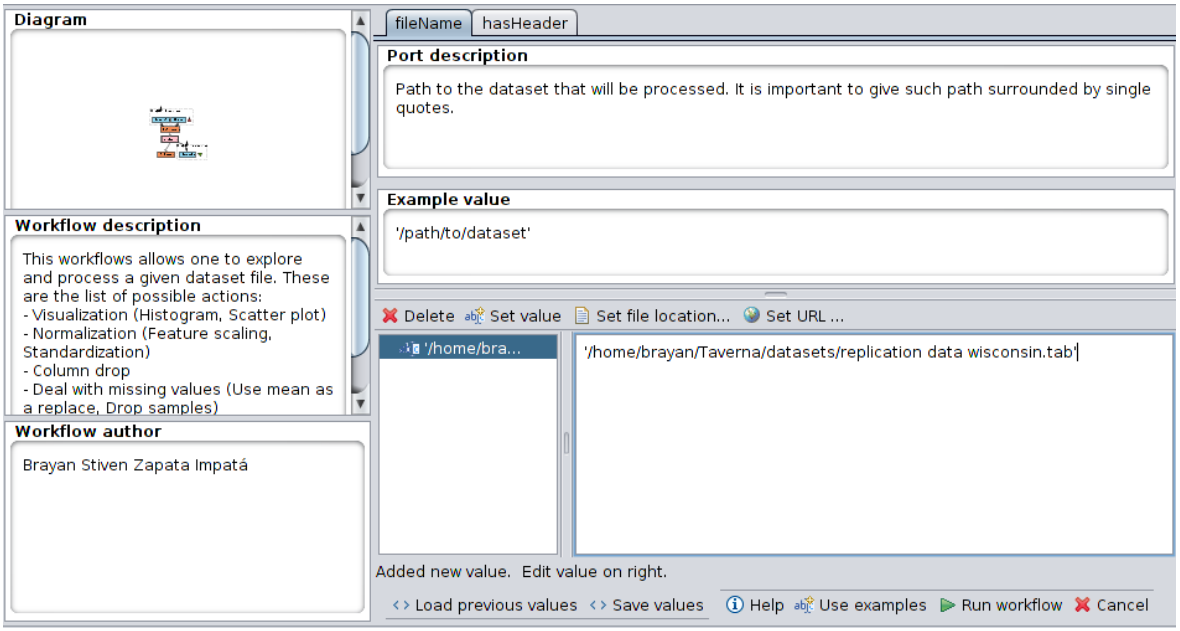


Figura 3.14: Entrada de datos en el puerto *fileName* para lanzar *dataExploration*.

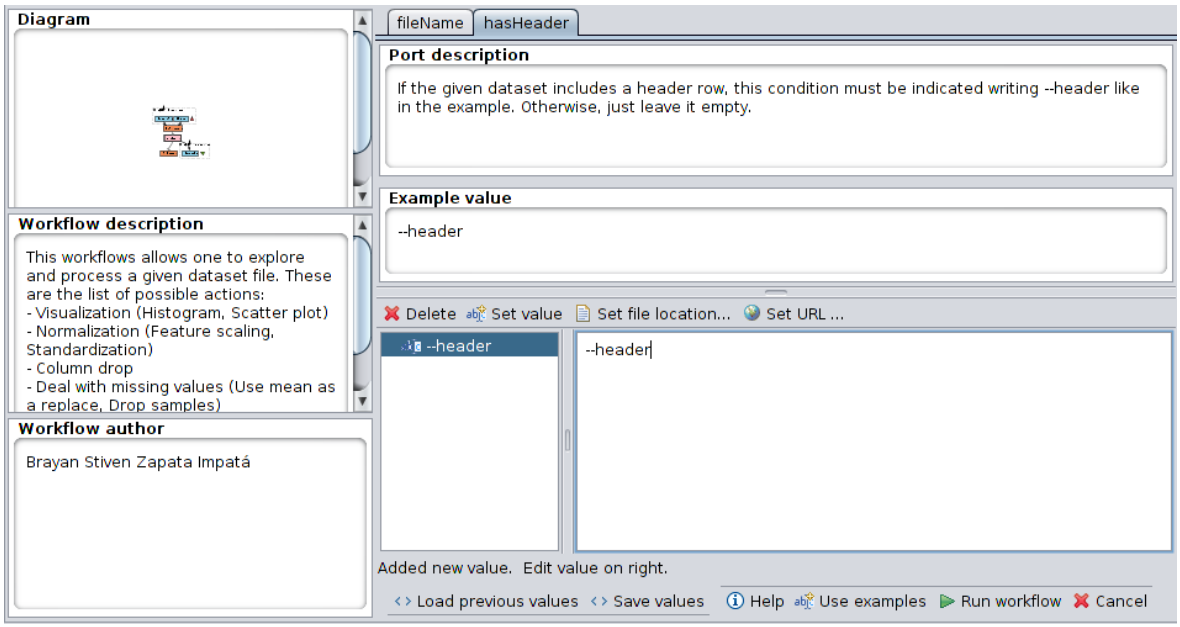


Figura 3.15: Entrada de datos en el puerto *hasHeader* para lanzar *dataExploration*.

Una vez arranca el sistema llegaremos a la pantalla mostrada en la figura 3.16. En esta ventana tendremos siempre arriba una descripción de los datos de entrada para disponer rápidamente de una serie de estadísticas para cada atributo que ayuden a su preprocesado. Estas estadísticas son: conteo, media, desviación estándar, valor mínimo, perceptiles (25 %, 50 %, 75 %) y valor máximo.

Las acciones que se han implementado son técnicas básicas de visualización o de preprocesado. Se han elegido estas para cubrir unas necesidades primarias en este tipo de proyectos y no se han abarcado más de momento para no descentrarnos del objetivo del proyecto. Estas opciones son:

1. **Visualización:** tenemos la posibilidad de dibujar un histograma o una nube de puntos.

2. **Normalización:** se ha implementado la normalización por *feature scaling* y por *standardization*.

3. **Borrado de columna:** se puede eliminar del conjunto una columna.

4. **Manejo de datos perdidos:** podemos sustituirlos por la media del atributo o borrar las muestras con datos perdidos.

5. **Conversión de etiquetas:** es posible pasar los valores de la variable objetivo a una enumeración.

6. **Guardado:** permite guardar el conjunto preprocesado en un nuevo fichero.

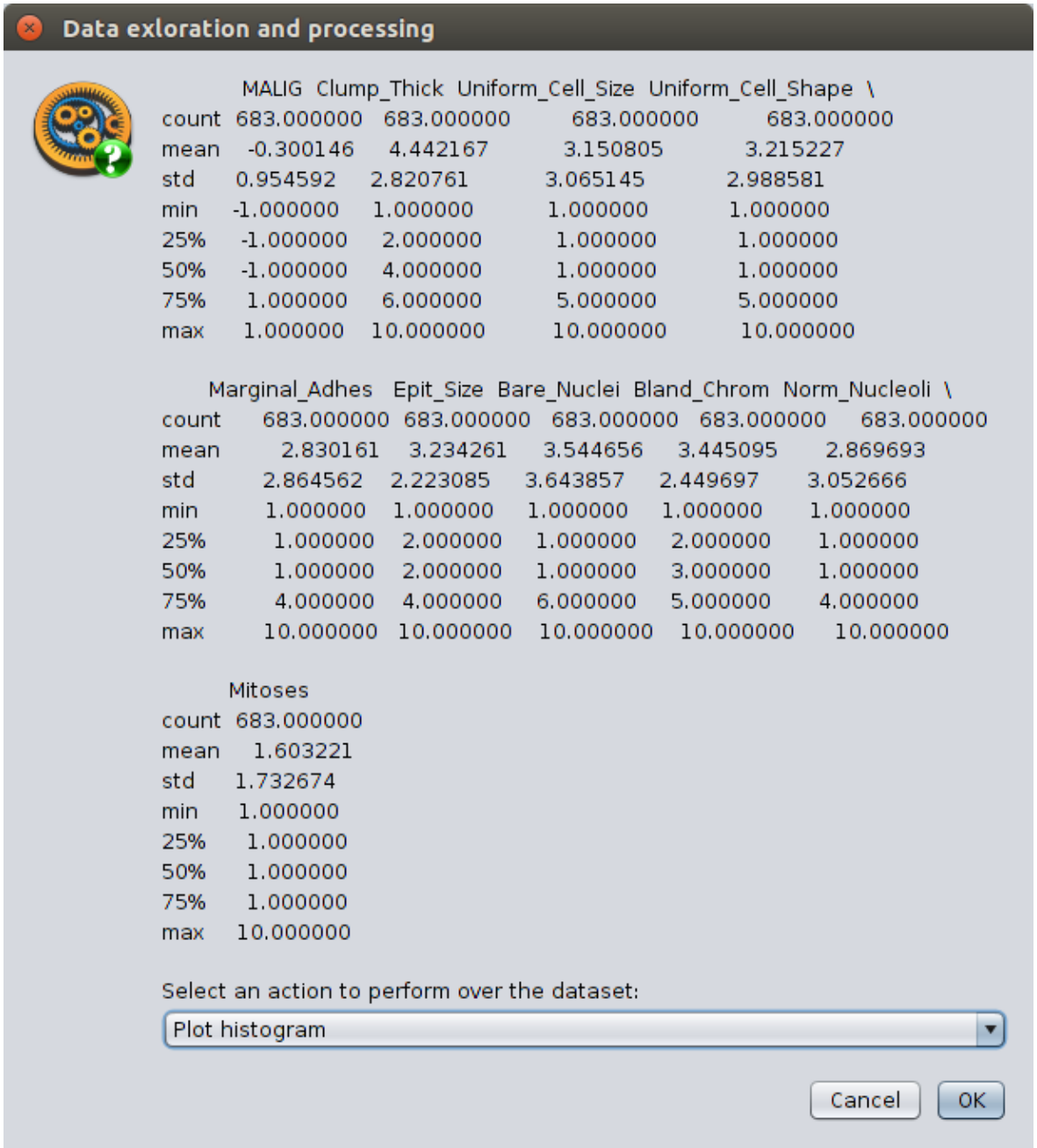


Figura 3.16: Pantalla principal de *dataExploration* con descripción del conjunto y selección de acciones a realizar.

Visualización

La primera posibilidad de visualización es la de dibujar una figura con el histograma de una columna. Su flujo en Taverna se presenta en la figura 3.17. Este básicamente representa la necesidad de indicar una columna del conjunto. Con ello, el sistema será capaz de dibujar el histograma.

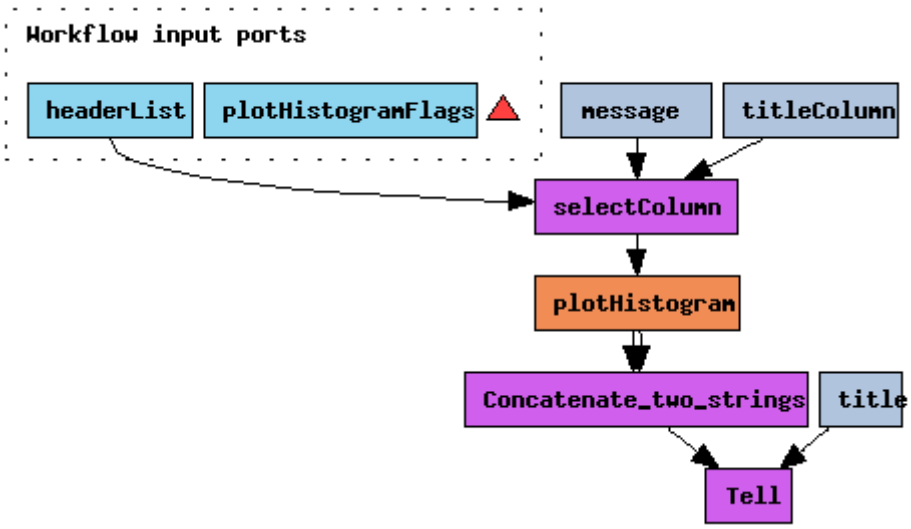


Figura 3.17: Diagrama con el flujo del proceso de dibujado de un histograma.

En la figura 3.18 se puede ver la ventana que nos ofrece la herramienta propuesta para seleccionar de un listado la columna a dibujar

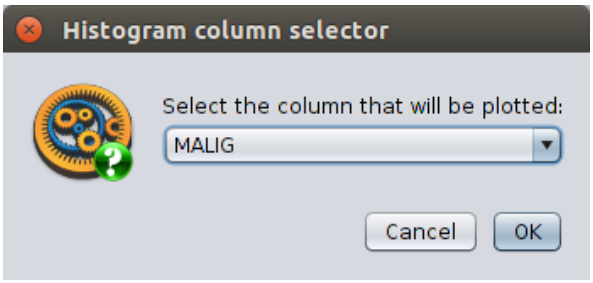


Figura 3.18: Ventana de selección de columna a usar para representar el histograma.

Una vez se selecciona la columna, el sistema ejecuta el código de Python necesario para crear la figura vista en 3.19 con el histograma. Esta figura es interactiva de manera que se puede desplazar sus ejes, reiniciarla o exportarla a un archivo de imagen. Además, mientras permanezca abierta el sistema completo estará en espera.

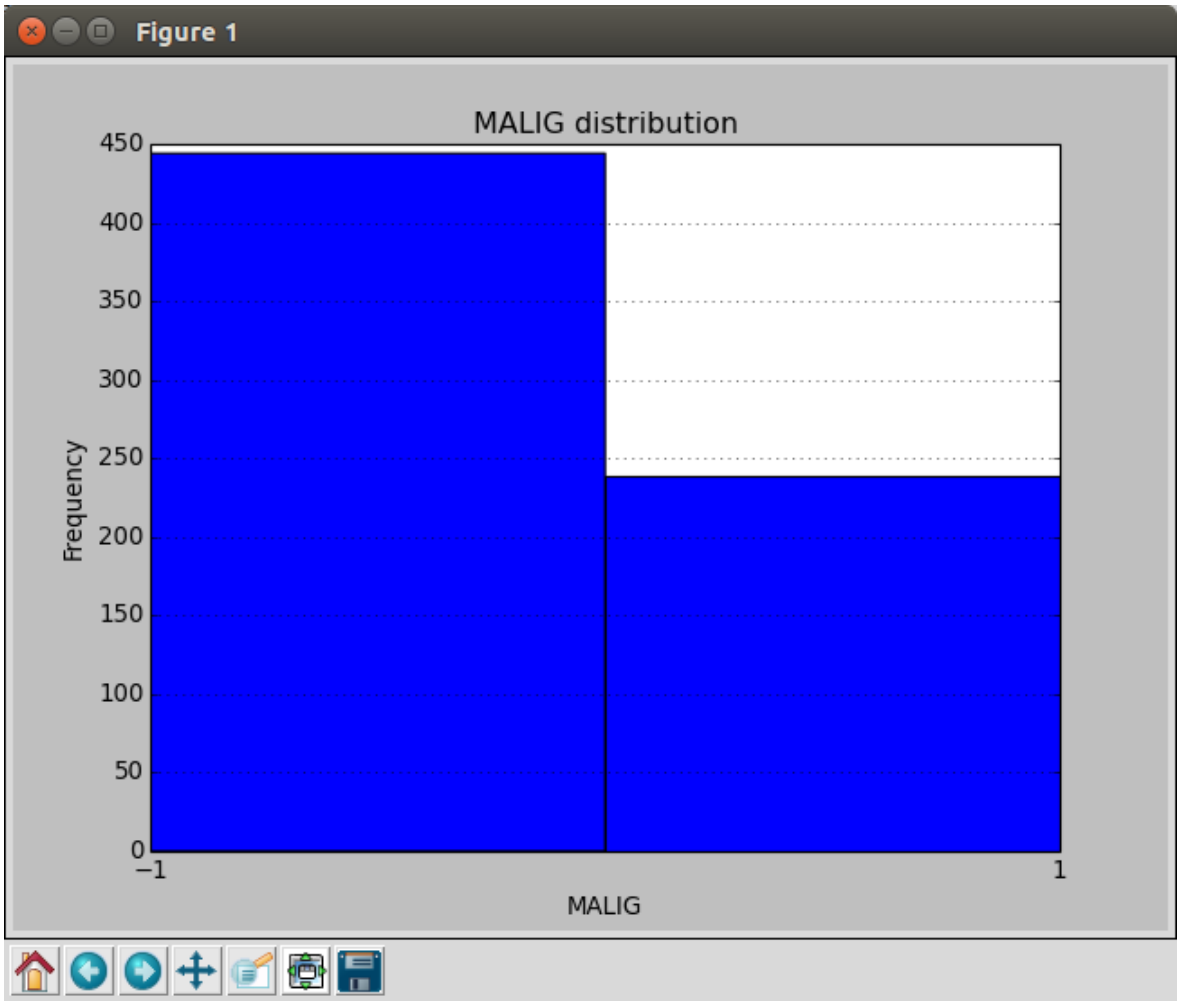


Figura 3.19: Ejemplo de histograma que la herramienta propuesta puede dibujar.

En cuanto cerremos la figura del histograma, la ejecución en Taverna continuará para indicarnos que el procesado ha sido correcto con el mensaje visto en la figura 3.20. Si hubiese un error, este se mostraría en este mensaje. Este mensaje se mostrará también al finalizar cualquier otra acción.

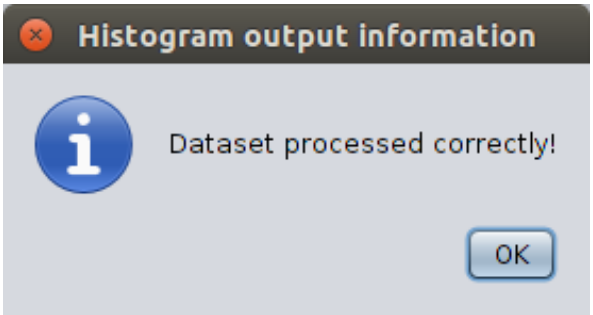


Figura 3.20: Mensaje de confirmación de finalización del histograma.

Otra posibilidad de visualización es la presentada con el diagrama de la figura 3.21. Esta figura representa el flujo del proceso que dibuja una nube de puntos dadas dos columnas y una tercera para colorearlas.

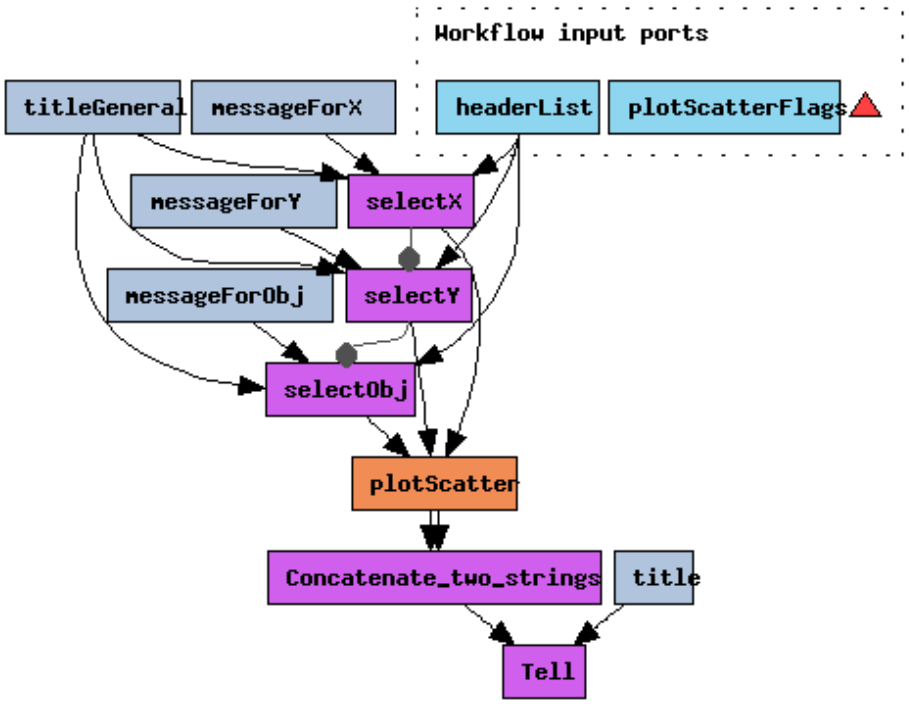


Figura 3.21: Diagrama con el flujo del proceso de dibujado de una nube de puntos.

Este proceso consiste en la selección de una columna para el eje X, otra columna para el eje Y y una última columna para colorear los puntos. Las ventanas mostradas en la figura 3.22 se suceden para conseguir esta selección.

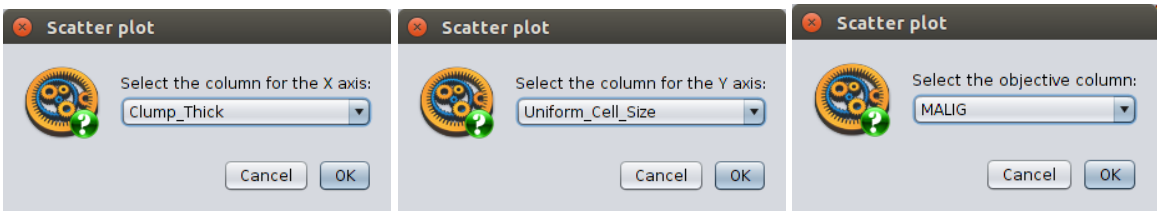


Figura 3.22: Ventanas de selección de columnas a usar para representar la nube de puntos.

Con todo ello, al final se consigue una figura que representa los datos elegidos como la presentada en 3.23. Esta ventana, como sucedía con la del histograma, es interactiva y tendrá al sistema en espera mientras permanezca abierta.

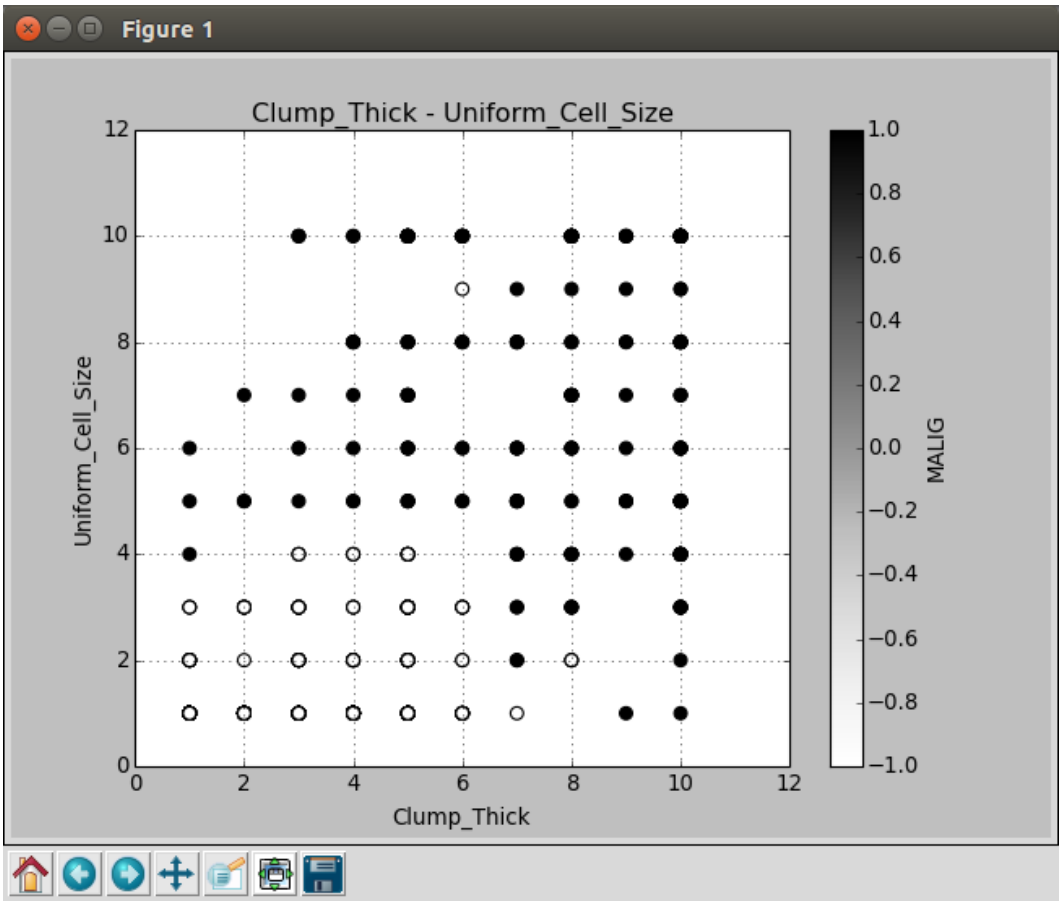


Figura 3.23

Normalización

Un caso muy típico en un proyecto donde se desea entrenar un modelo de aprendizaje automático es el de realizar una normalización de los datos. Esto suele suceder porque los atributos del conjunto poseen escalas muy desiguales entre sí y pueden conllevar a que el algoritmo de aprendizaje sea subjetivo. En nuestro sistema se han implementado dos conocidas técnicas: *feature scaling* y *standardization*.

La normalización por *feature scaling* deja los datos en el rango $[0, 1]$ utilizando la fórmula de la ecuación (3.1), donde x representa el valor de una columna determinada y x' el nuevo valor. Así $min(x)$ y $max(x)$ representan el valor mínimo y máximo respectivamente de dicha columna.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

Su implementación en este proyecto se presenta con el diagrama de flujo de Taverna de la figura 3.24. Este proceso ejecuta el código de Python que realiza la normalización y deja el conjunto con la proporción mostrada en la figura 3.25.

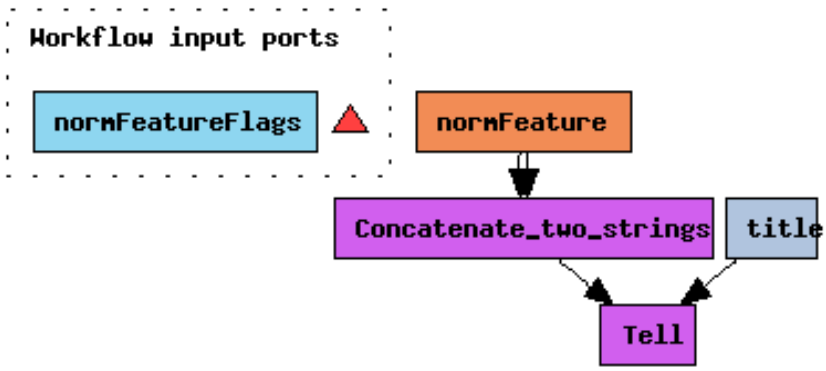


Figura 3.24: Diagrama con el flujo de la normalización por *feature scaling*.

	MALIG	Clump_Thick	Uniform_Cell_Size	Uniform_Cell_Shape
count	683.000000	683.000000	683.000000	683.000000
mean	0.349927	0.382463	0.238978	0.246136
std	0.477296	0.313418	0.340572	0.332065
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.111111	0.000000	0.000000
50%	0.000000	0.333333	0.000000	0.000000
75%	1.000000	0.555556	0.444444	0.444444
max	1.000000	1.000000	1.000000	1.000000

	Marginal_Adhes	Epit_Size	Bare_Nuclei	Bland_Chrom	Norm_Nucleoli
count	683.000000	683.000000	683.000000	683.000000	683.000000
mean	0.203351	0.248251	0.282740	0.271677	0.207744
std	0.318285	0.247009	0.404873	0.272189	0.339185
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.111111	0.000000	0.111111	0.000000
50%	0.000000	0.111111	0.000000	0.222222	0.000000
75%	0.333333	0.333333	0.555556	0.444444	0.333333
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Mitoses
count	683.000000
mean	0.067025
std	0.192519
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Figura 3.25: Descripción del conjunto tras la normalización por *feature scaling*.

Para la normalización por *standardization* se utiliza la ecuación (3.2), donde μ representa el valor medio de la columna y σ la desviación estándar. Como resultado, las nuevas columnas x' tendrán una media de 0 y desviación de 1.

$$x' = \frac{x - \mu}{\sigma} \tag{3.2}$$

La figura 3.26 presenta su diagrama en la implementación realizada. Este proceso ejecuta el código Python que deja el conjunto en el estado visto en la figura 3.27.

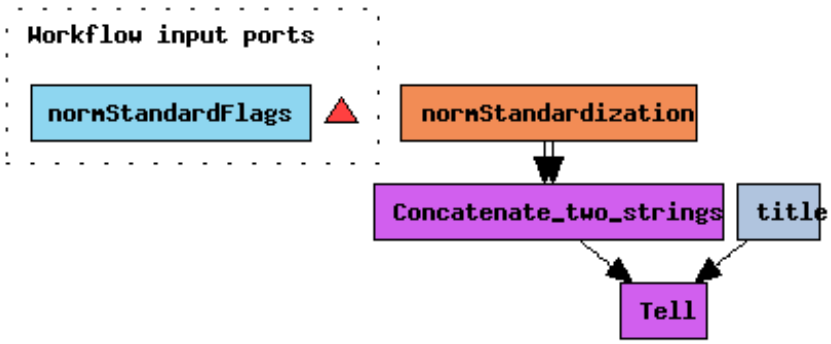


Figura 3.26: Diagrama con el flujo de la normalización por *standardization*.

	MALIG	Clump_Thick	Uniform_Cell_Size	Uniform_Cell_Shape
count	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02
mean	7.145376e-13	2.386612e-13	4.287184e-13	2.479929e-13
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-7.331440e-01	-1.220297e+00	-7.016978e-01	-7.412304e-01
25%	-7.331440e-01	-8.657829e-01	-7.016978e-01	-7.412304e-01
50%	-7.331440e-01	-1.567545e-01	-7.016978e-01	-7.412304e-01
75%	1.361991e+00	5.522740e-01	6.032977e-01	5.971975e-01
max	1.361991e+00	1.970331e+00	2.234542e+00	2.270232e+00

	Marginal_Adhes	Epit_Size	Bare_Nuclei	Bland_Chrom
count	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02
mean	4.808647e-13	-3.089560e-13	7.233596e-13	3.806761e-13
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-6.388973e-01	-1.005027e+00	-6.983413e-01	-9.981216e-01
25%	-6.388973e-01	-5.552016e-01	-6.983413e-01	-5.899078e-01
50%	-6.388973e-01	-5.552016e-01	-6.983413e-01	-1.816940e-01
75%	4.083832e-01	3.444489e-01	6.738310e-01	6.347336e-01
max	2.502944e+00	3.043400e+00	1.771569e+00	2.675803e+00

	Norm_Nucleoli	Mitoses
count	6.830000e+02	6.830000e+02
mean	-3.616382e-13	4.201877e-13
std	1.000000e+00	1.000000e+00
min	-6.124785e-01	-3.481446e-01
25%	-6.124785e-01	-3.481446e-01
50%	-6.124785e-01	-3.481446e-01
75%	3.702689e-01	-3.481446e-01
max	2.335764e+00	4.846139e+00

Figura 3.27: Descripción del conjunto tras la normalización por *standardization*.

Borrado de columnas

Existe la posibilidad de que tengamos una columna que no sea de nuestro interés en el conjunto, ya sea porque es una columna de datos no numéricos que no queremos utilizar o porque sea una secuencia de números que no aporta nada. Sea cual sea el motivo, en el sistema propuesto se contempla la opción de eliminar una columna del conjunto. Su implementación se presenta en la figura 3.28.

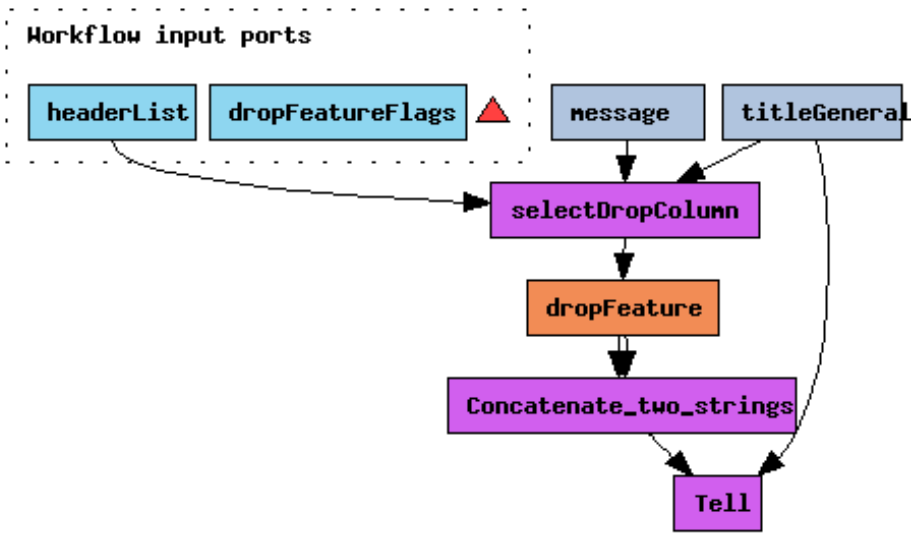


Figura 3.28: Diagrama con el flujo del proceso de borrado de una columna.

Para poder borrar una columna, el sistema nos pedirá que le indiquemos cuál columna queremos eliminar con una ventana como la presentada en la figura 3.29.

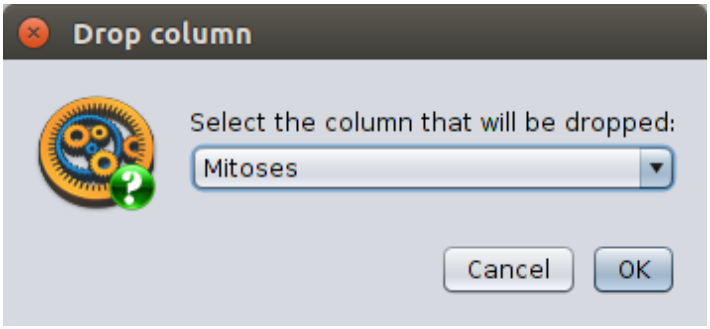


Figura 3.29: Ventana de selección de columna a borrar del conjunto.

Una vez seleccionada la columna, el proceso pasa a ejecutar el código necesario para dejar el conjunto cargado sin dicha columna. Así, al volver a la pantalla de

inicio veremos que ya no está. En este ejemplo de uso hemos borrado la última columna llamada `Mitoses`, que como se muestra en la figura 3.30 ya no aparece en el conjunto.

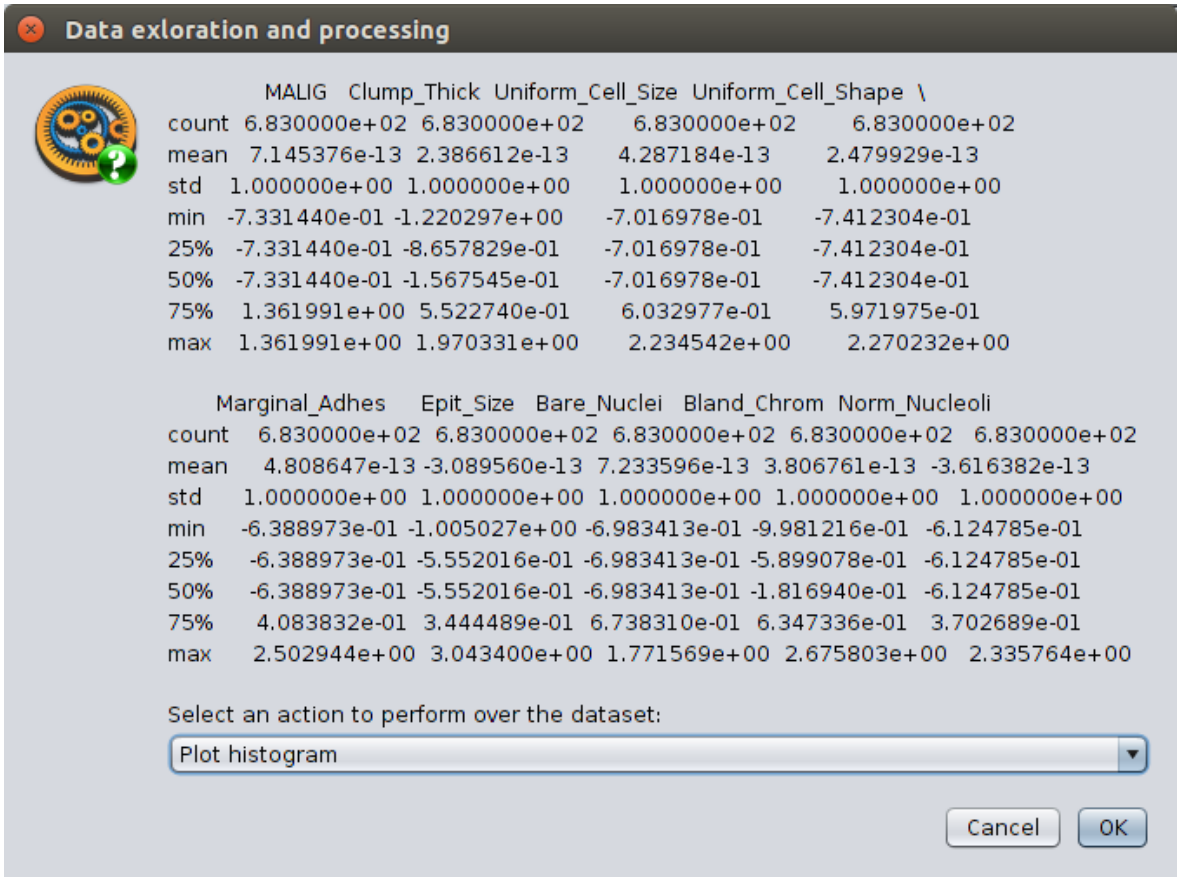


Figura 3.30: Descripción del conjunto tras el borrado de la columna seleccionada.

Manejo de datos perdidos

Un caso muy común en proyectos de minería es que alguna muestra no tenga algún valor informado. El manejo de estos datos es complicado y se expande en todo un campo aún en investigación para ver qué acciones son las más correctas. En el presente trabajo se han considerado dos opciones básicas para no desviarnos del objetivo del proyecto.

Para escenificar el uso de esta funcionalidad vamos a considerar que cargamos las 699 muestras del conjunto de datos inicial de la UCI, nombrado en la introducción del capítulo. Al cargarlo tenemos la descripción del conjunto vista en la figura

3.31 donde vemos que el atributo **6** tiene 683 muestras, 16 menos que el resto de atributos. Esto es así porque existen 16 muestras en el conjunto que no tienen valor en este atributo.

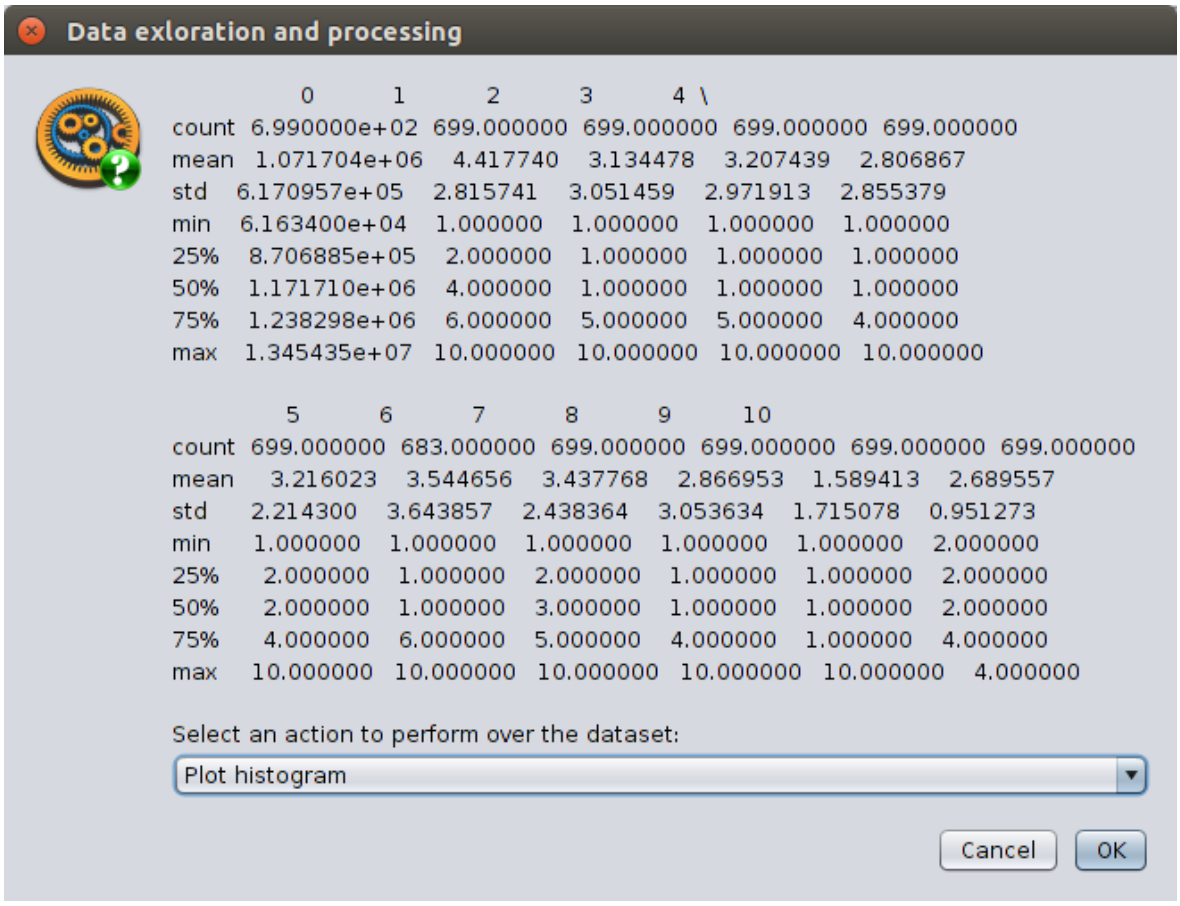


Figura 3.31: Descripción del conjunto inicial que posee valores perdidos en la columna 6.

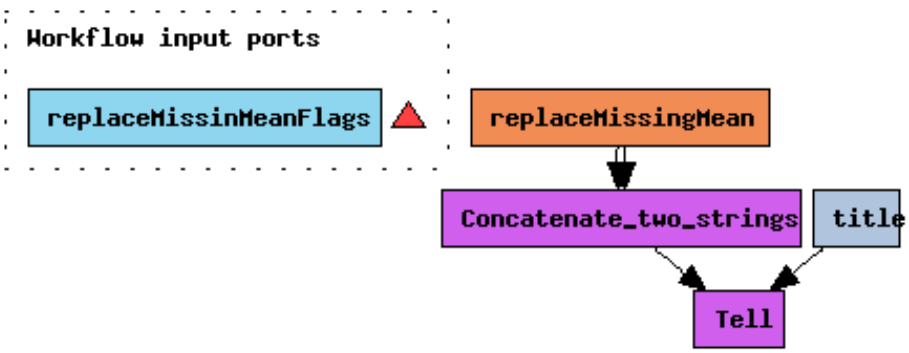


Figura 3.32: Diagrama con el flujo del proceso de remplazo de valores perdidos.

La primera técnica implementada consiste en la inserción de valores para completar estas muestras. En concreto, el dato que se inserta es el valor medio que poseen las muestras del conjunto para el atributo perdido. Su implementación en el flujo de Taverna se muestra en la figura 3.32. Tras su ejecución sobre el conjunto cargado, podemos ver como el atributo 6 pasa a tener 699 muestras al igual que el resto (ver fig. 3.33).

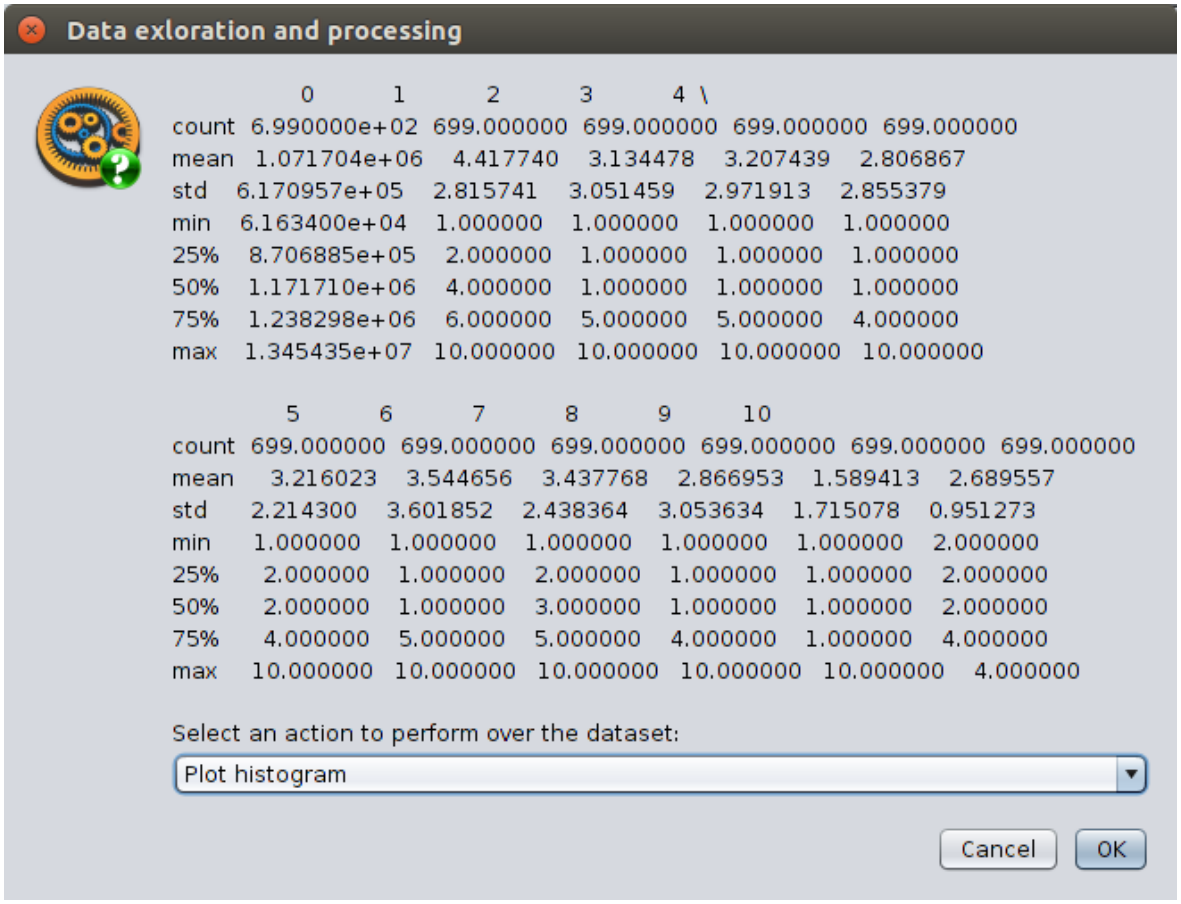


Figura 3.33: Descripción del conjunto con valores perdidos tras su procesado por remplazo. Ahora la columna 6 tiene nuevos datos e iguala en muestras al resto de atributos.

La otra técnica implementada consiste en el borrado del conjunto de las muestras que tengan algún valor perdido entre sus atributos. En la figura 3.34 se muestra su flujo en Taverna y en la figura 3.35 se ve el resultado de su procesamiento. Como se puede comprobar, tras el borrado de muestras lo que tenemos es un conjunto más pequeño pues las 16 muestras que tenían valores perdidos se eliminan.

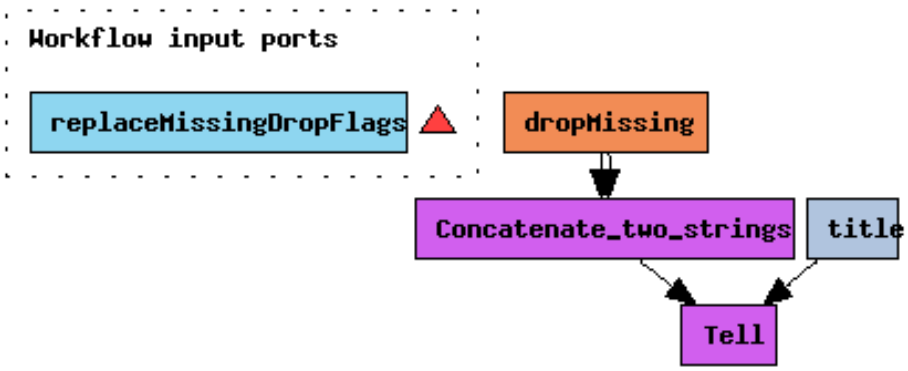


Figura 3.34: Diagrama con el flujo del proceso de borrado de muestras con valores perdidos.

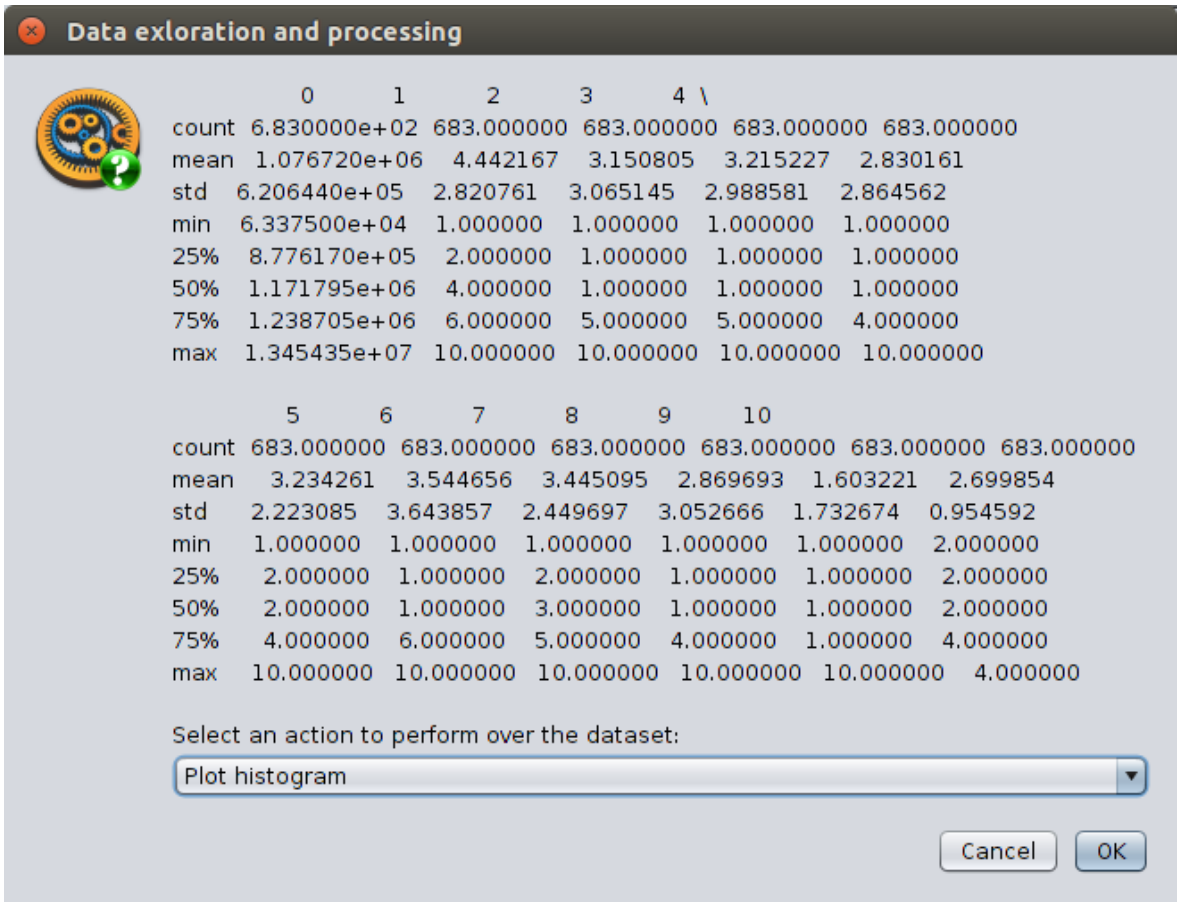


Figura 3.35: Descripción del conjunto con valores perdidos tras su procesado por borrado de muestras. Ahora el conjunto posee 683 muestras.

Conversión numérica de clases

En los conjuntos de datos utilizados para construir modelos clasificadores por aprendizaje supervisado siempre encontraremos una columna que etiqueta la clase

a la que pertenece la muestra. Para el correcto funcionamiento del programa que proponemos en este proyecto, es necesario que la variable dependiente (la que indica la clase de la muestra) sea numérica y vaya numerada a partir del 0.

Para poder corregir los conjuntos que no cumplan esta condición podemos utilizar esta función del programa *dataExploration* que lo único que nos pedirá es que indicamos el nombre de la columna con las clases. Su implementación se muestra en el *workflow* de la figura 3.36.

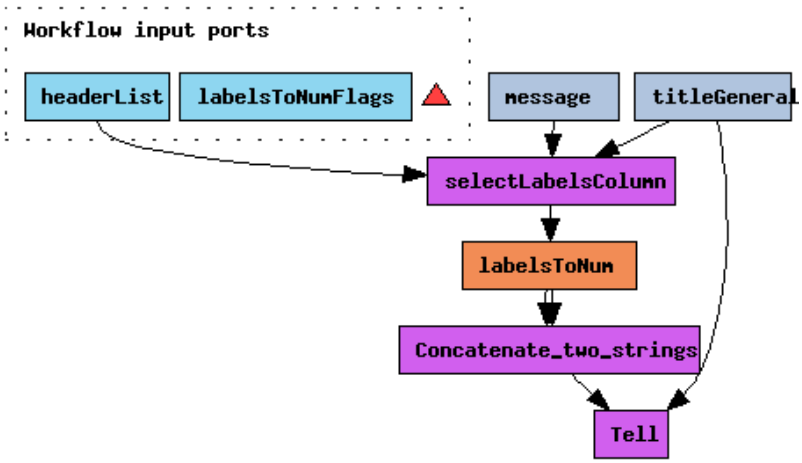


Figura 3.36: Diagrama con el flujo del proceso de conversión de clases a enumeraciones.

Por ejemplo, en el conjunto descargado tenemos la variable dependiente `MALIG`. En la figura 3.27 se ve como toma valores decimales tras el proceso de *standardization*. Podemos hacerla tomar los valores 0 o 1 con esta técnica. Para ello primero se indica la columna, como se ve en la figura 3.37.

Figura 3.37: Ventana de selección de columna con etiquetas.

Después de seleccionar la columna adecuada, el sistema la procesa para mapear sus valores de forma que queden en una numeración que comienza por 0 como se muestra en la figura 3.38. Esto viene a significar que si un conjunto tiene en su columna de etiquetas los valores 2 y 4, estas etiquetas pasarán a valer 0 y 1.

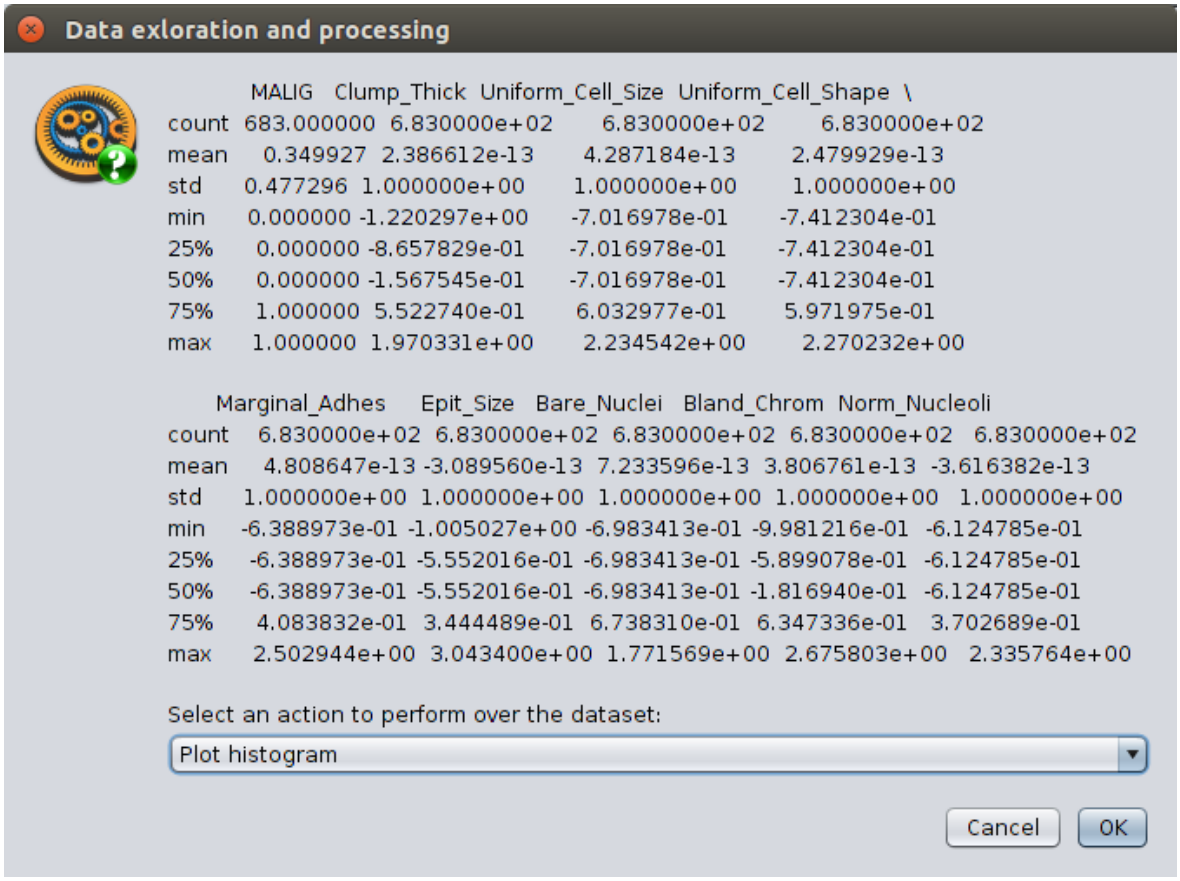


Figura 3.38: Descripción del conjunto tras la conversión de la columna de etiquetas *MALIG*.

Guardado

Finalmente, cuando tengamos el conjunto preprocesado hasta el punto que deseemos podemos guardarlo en un nuevo fichero. El diagrama de la figura 3.39 presenta el flujo de este proceso en la implementación de Taverna. Para realizar esta acción el sistema solicitará un nombre para el nuevo fichero, como se ve en la ventana de la figura 3.40. Con el fichero creado, este se guarda en una ruta configurada en el componente `saveNewFile`, que se confirma como se muestra en la figura 3.41.

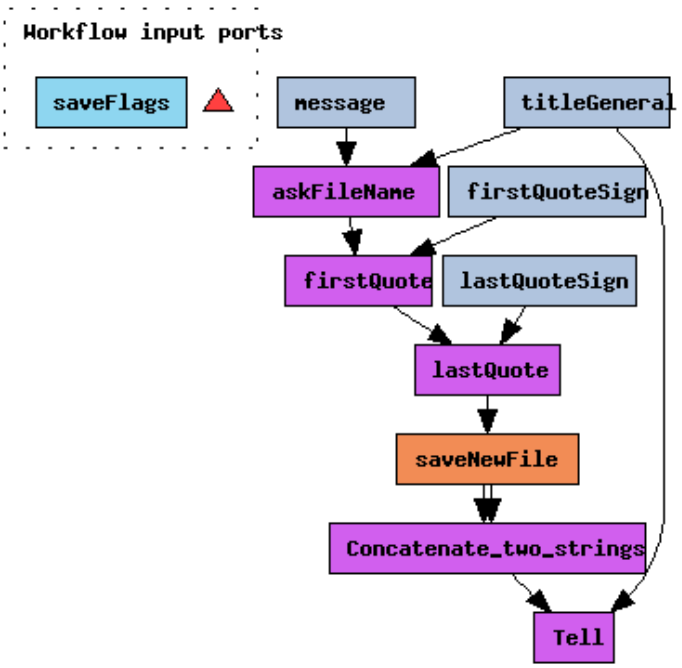


Figura 3.39: Diagrama con el flujo del proceso de guardar el conjunto procesado en un nuevo fichero.

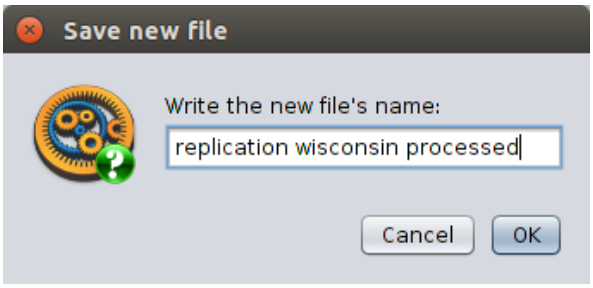


Figura 3.40: Ventana de solicitud del nombre del nuevo fichero a guardar.

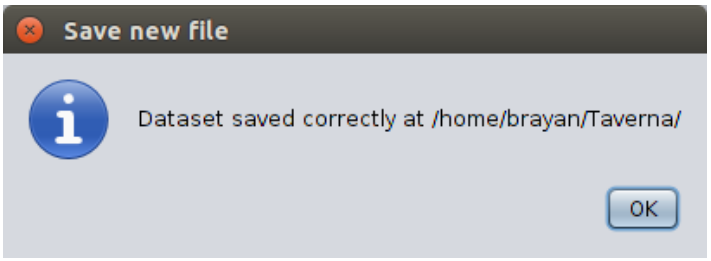


Figura 3.41: Mensaje de confirmación de la creación del nuevo fichero.

3.5. Aprendizaje automático

Para el entrenamiento de un modelo recomendador utilizamos el conjunto descargado `replication data wisconsin.tab`. Sin embargo, lo preprocesamos con el programa *dataExploration* y le realizamos una normalización por *feature scaling* para igualar las escalas de los campos y a su vez dejar la columna dependiente `MALIG` con valores 0 o 1. Al fichero resultante lo llamamos `replication wisconsin processed`.

La figura 3.42 presenta el diagrama de flujo de la implementación de Taverna. Esta implementación se basa en un script de Python, insertado en el servicio *modelBuild* del diagrama, que utiliza Tensorflow para entrenar un modelo llamado *softmax regression*, que es una generalización de la regresión logística, comúnmente usada para problemas multiclase.

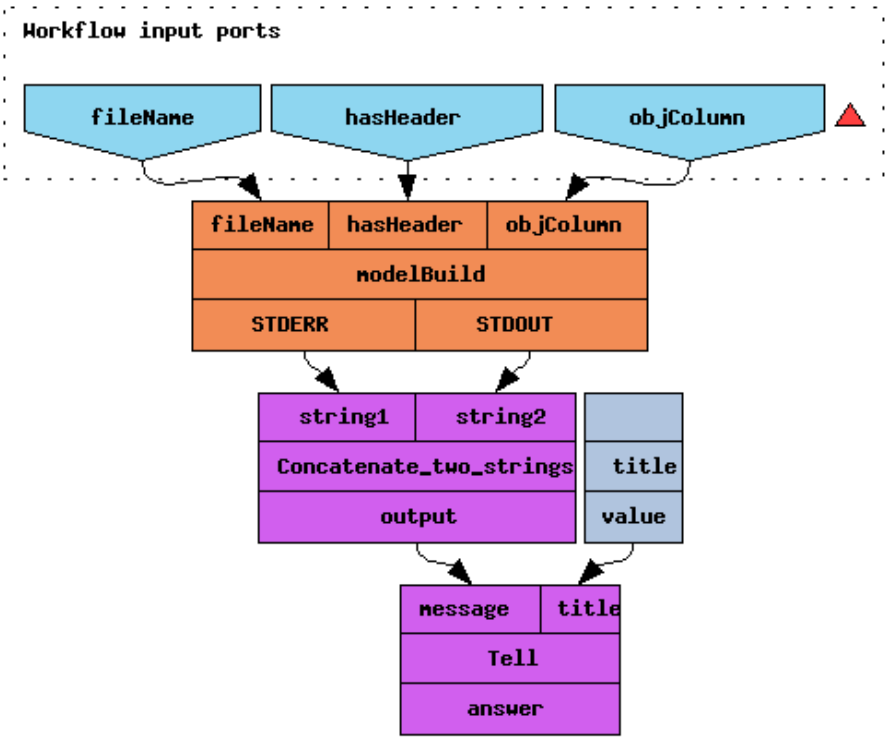


Figura 3.42: Diagrama con el flujo del proceso *modelBuild* en Taverna.

El proceso de entrenamiento consiste en encontrar los valores del vector θ de forma que se minimice el error de la hipótesis definida en (3.3), que debe ser igual a la etiqueta de la muestra x en el conjunto de entrenamiento.

$$h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}} \tag{3.3}$$

El resultado de una predicción con este modelo es una probabilidad de pertenencia a cada categoría del problema. Dada su capacidad de procesar problemas multiclase, se ha elegido para implementarse en este proyecto de forma que el sistema propuesto pueda trabajar con una mayor diversidad de problemas sin necesidad de modificaciones.

Las entradas que necesita el flujo para ejecutarse son:

- **fileName:** ruta al fichero a utilizar para realizar el entrenamiento, que debe tener una columna de clase con números empezando por 0.
- **hasHeader:** en caso de que el archivo tenga una línea de cabecera con los nombres de las columnas, se debe indicar el valor `--header`, sino se da un valor vacío y el programa creará una cabecera numérica empezando por 0.
- **objColumn:** nombre exacto que posee la columna de etiquetas, en caso de no tener cabecera el fichero entonces se debe indicar su posición en el conjunto empezando por 0.

Con todo esto, lanzamos una ejecución utilizando los valores vistos en las figuras 3.43, 3.44 y 3.45.

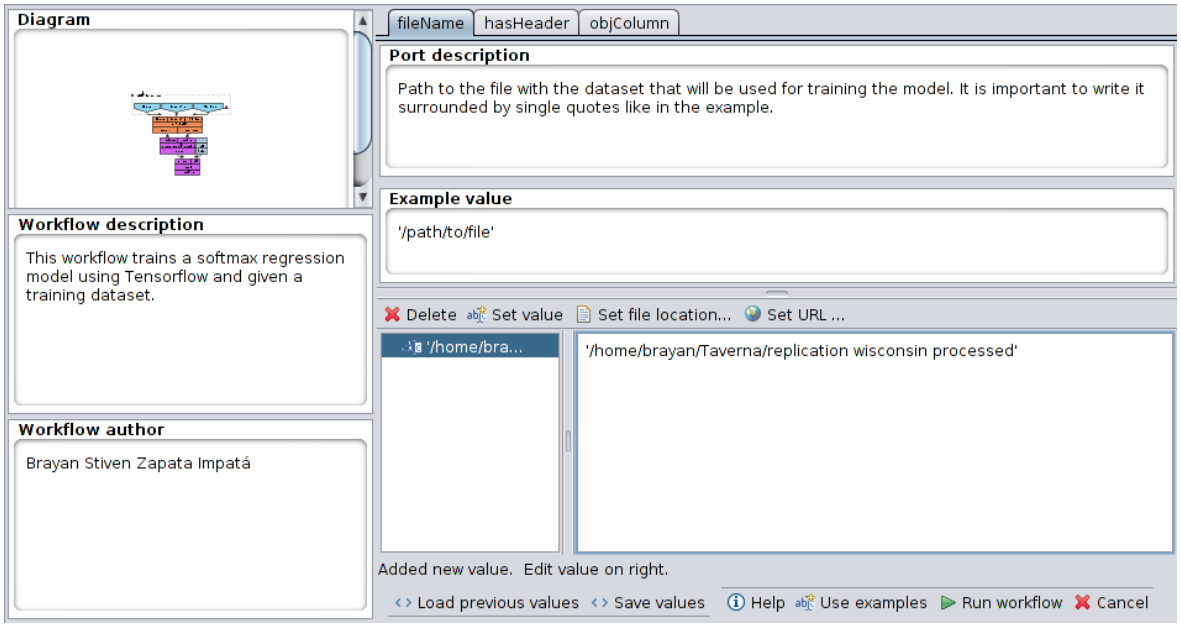


Figura 3.43: Ejemplo de uso del puerto de entrada *fileName* de *modelBuild*.

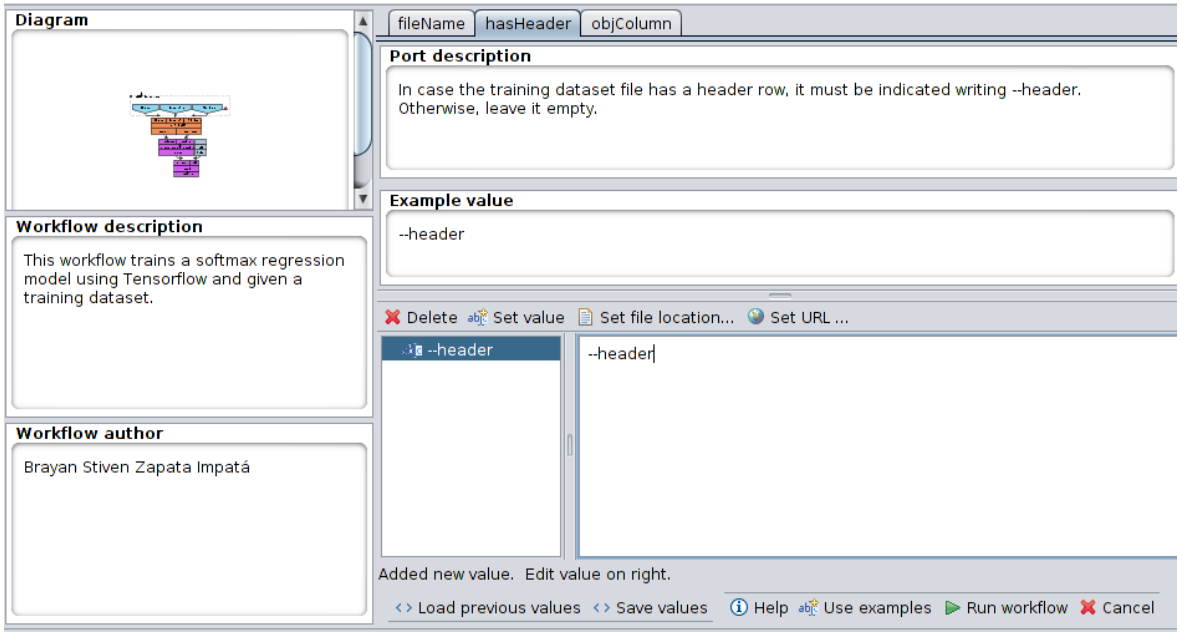


Figura 3.44: Ejemplo de uso del puerto de entrada *hasHeader* de *modelBuild*.

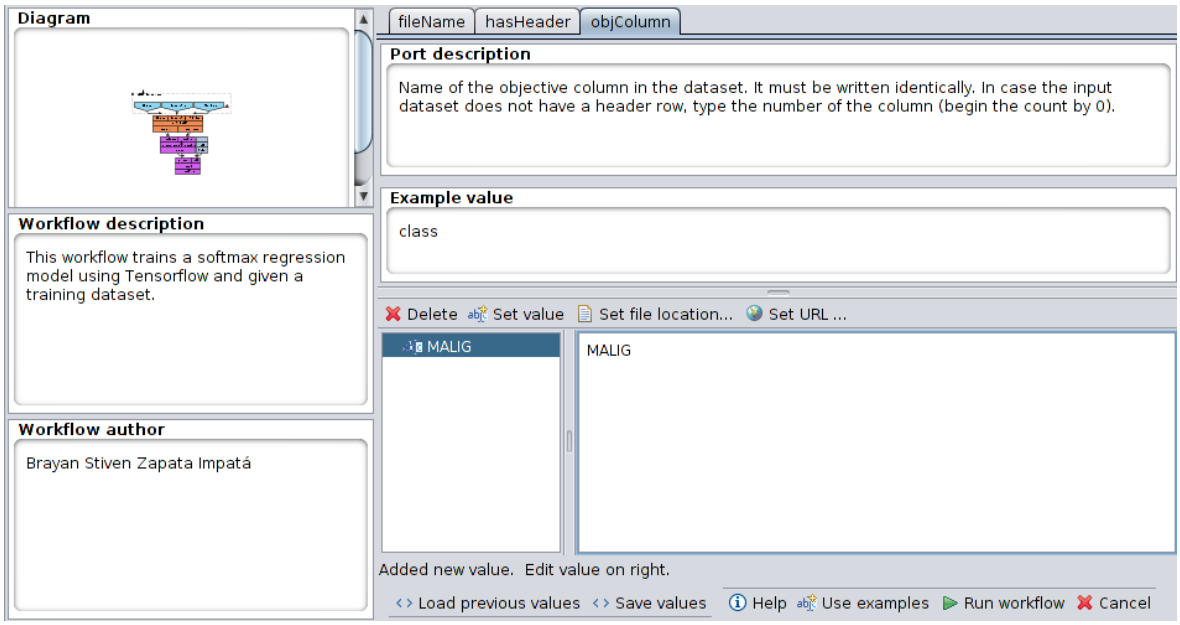


Figura 3.45: Ejemplo de uso del puerto de entrada *objColumn* de *modelBuild*.

Una vez se lanza la ejecución, empezará el entrenamiento de Tensorflow. El tiempo que este proceso toma varía principalmente según el rendimiento del equipo donde se ejecuta el programa, el número de muestras incluidas en el conjunto así como la cantidad de atributos que estas poseen. Cuando termina, tendremos una ventana informativa como la mostrada en la figura 3.46.

Como se ve en esta figura, Tensorflow emite una serie de mensajes cuando arranca en una máquina que utiliza su implementación con CUDA para informar al usuario sobre la GPU a utilizar. En lo referente a la información que devuelve el programa, podremos ver cuáles son los valores de θ aprendidos así como los del término conocido como *bias*. Habrá una columna por cada clase del problema.

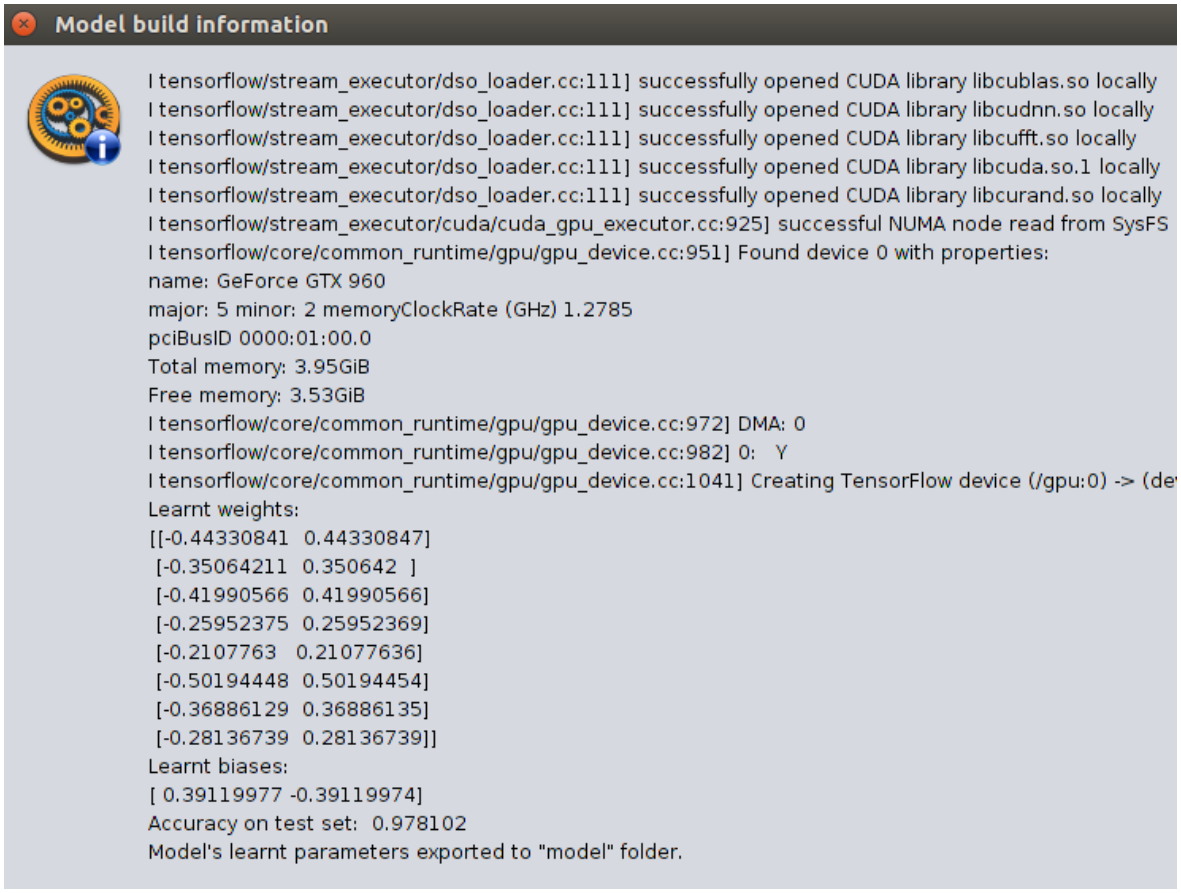


Figura 3.46: Información que devuelve el programa *modelBuild* tras su procesamiento.

Con estos parámetros aprendidos el sistema realiza una prueba sobre un conjunto que ha guardado y no ha utilizado para el entrenamiento. Del total de muestras

que se han cargado del fichero, el 80 % se utilizan para entrenar mientras que el restante 20 % se dejan para probar el rendimiento del modelo. En este caso el modelo entrenado consigue un porcentaje de aciertos del 97.8102 %.

Finalmente, el sistema crea una serie de ficheros con los datos del modelo. Estos ficheros se almacenan en una ruta configurada dentro del script de Python. En la figura 3.47 tenemos una muestra de estos ficheros:

- **modelVariables.txt:** este documento guarda los nombres de las variables independientes que se han utilizado para entrenar el modelo, manteniendo también el orden en el que se han usado pues es importante a la hora de reutilizar el modelo.
- **modelWeights.txt:** en este documento se almacenan los pesos o valores del parámetro θ aprendidos de una forma que son fácilmente reaprovechables.
- **modelBiases.txt:** al igual que con los pesos, se almacenan los valores de *bias* de forma que se pueda reaprovechar el modelo.

```
brayan@orionux:~/Taverna$ cat modelVariables.txt
Clump_Thick
Uniform_Cell_Size
Uniform_Cell_Shape
Marginal_Adhes
Epit_Size
Bare_Nuclei
Bland_Chrom
Norm_Nucleoli
brayan@orionux:~/Taverna$ cat modelWeights.txt
-4.433084130287170410e-01 4.433084726333618164e-01
-3.506421148777008057e-01 3.506419956684112549e-01
-4.199056625366210938e-01 4.199056625366210938e-01
-2.595237493515014648e-01 2.595236897468566895e-01
-2.107762992382049561e-01 2.107763588428497314e-01
-5.019444823265075684e-01 5.019445419311523438e-01
-3.688612878322601318e-01 3.688613474369049072e-01
-2.813673913478851318e-01 2.813673913478851318e-01
brayan@orionux:~/Taverna$ cat modelBiases.txt
3.911997675895690918e-01
-3.911997377872467041e-01
brayan@orionux:~/Taverna$
```

Figura 3.47: Contenido de los ficheros que almacenan los datos necesarios para reaprovechar el modelo entrenado.

Además de los ficheros con los datos del modelo entrenado, el programa *modelBuild.py* también genera un directorio con un registro de Tensorflow que puede ser visualizado en su herramienta Tensorboard. Este se encuentra en un directorio lla-

mado `tmp_log_board` y cada ejecución del programa creará su propia carpeta con el momento de ejecución como nombre.

Para poder visualizar este registro tan solo es necesario lanzar el comando `tensorboard --logdir tmp_log_board` como se muestra en la figura 3.48.

```
brayan@orionux:~/TFM$ tensorboard --logdir tmp_log_board/
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library
 libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library
 libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library
 libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library
 libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library
 libcurand.so locally
Starting TensorBoard 29 on port 6006
(You can navigate to http://127.0.1.1:6006)
```

Figura 3.48: Lanzamiento de Tensorboard.

Tras el lanzamiento se iniciará un servidor web al que podemos acceder a través de un navegador introduciendo la dirección `http://127.0.0.1:6006`. Al abrir esta dirección entraremos en Tensorboard y lo primero que veremos será la pestaña *Events*, donde podemos comprobar la evolución del acierto y del coste de la función que se minimiza para encontrar los valores de θ durante el entrenamiento. Esta página se muestra en la figura 3.49.

En la pestaña *Graphs* podemos ver el grafo que define el proceso por el cual se construye el modelo de aprendizaje y se entrena. Este grafo se muestra en la figura 3.50. Lo que este gráfico representa es que se definen unas entradas donde se introduce una matriz con las muestras y un vector con las clases de esas muestras. Por otro lado se define la función que lanzará las predicciones, que dependerá de los parámetros *weights* y *biases* que son los que se van a calcular.

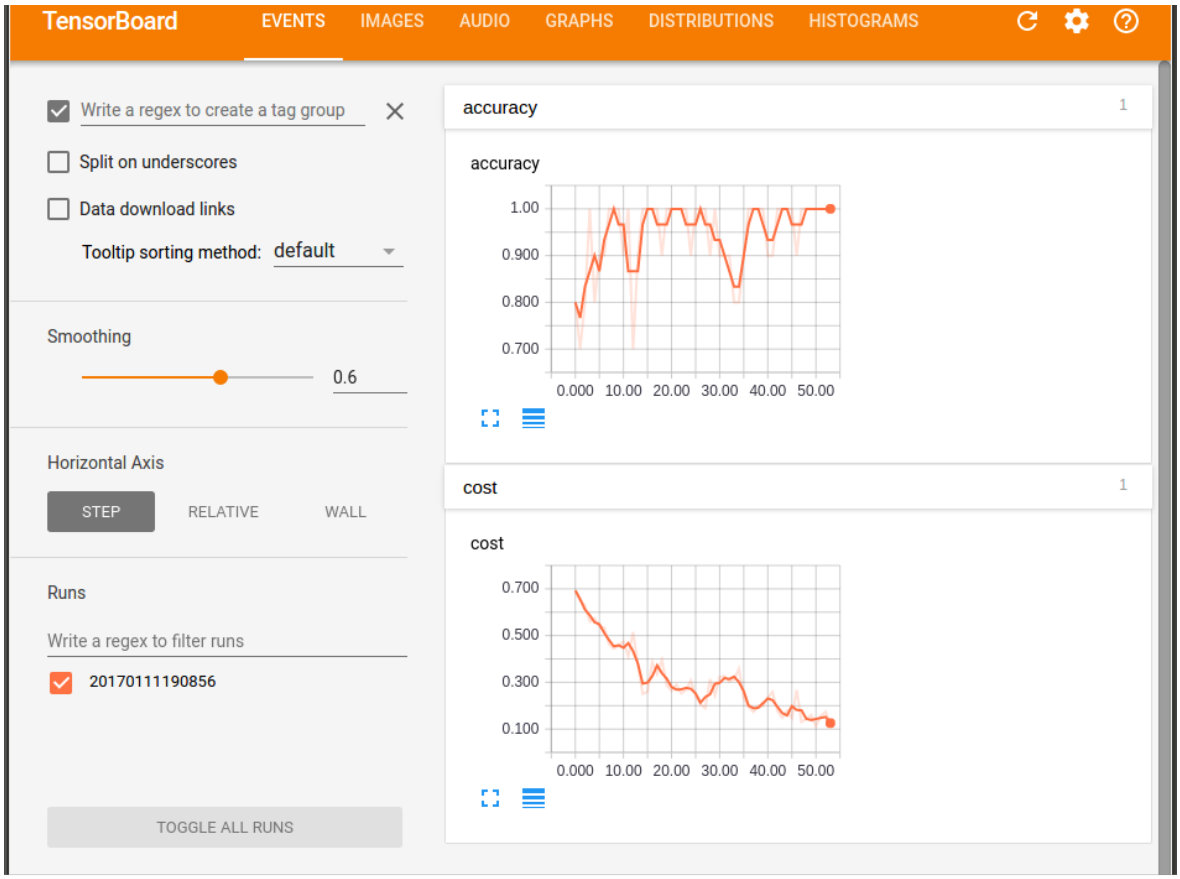


Figura 3.49: Pestaña de eventos en Tensorboard.

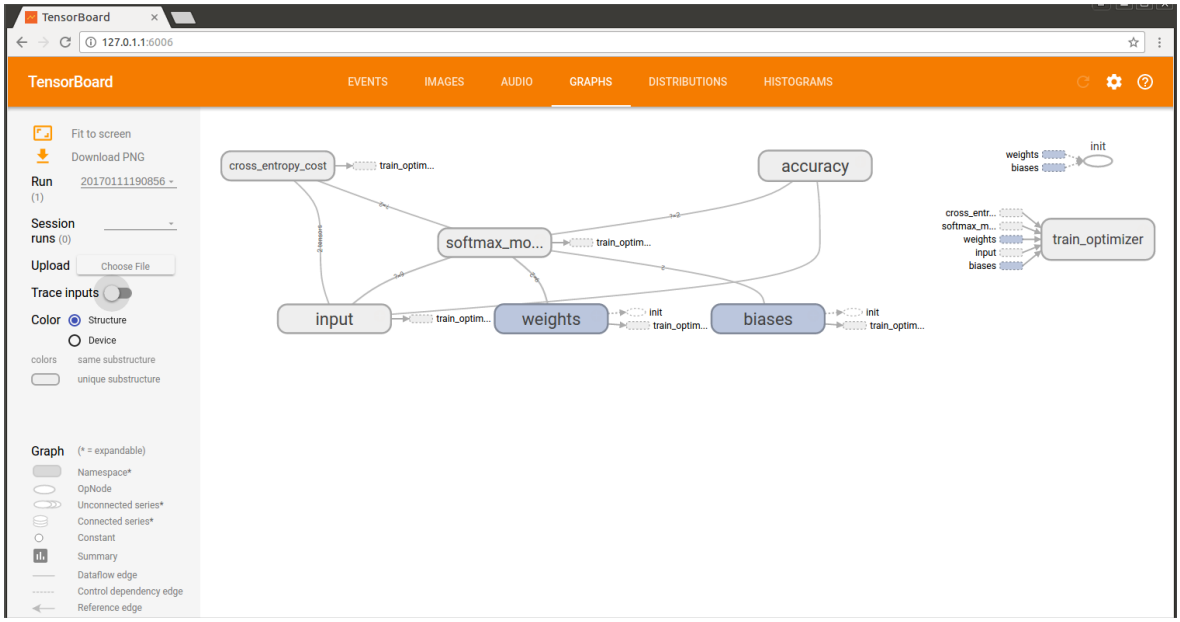


Figura 3.50: Panel de ayuda del programa *modelBuild.py*.

Esta funcionalidad y datos de Tensorboard están incluidos en la propuesta ya que se considera importante poder ver cómo evoluciona el acierto por si, por ejemplo, probamos con distintos subconjuntos de un mismo fichero de muestras que varían en características o preprocesado.

Además, con ella también se pretende dar pié a compartir y comparar modelos de entrenamiento entre los usuarios expertos que modifiquen este código, de forma que se pueda crear un repositorio de modelos donde abrir a otros investigadores el gráfico que representa cómo se han obtenido los parámetros almacenados en los ficheros `modelWeights` y `modelBiases`, pudiendo indicar también las variables utilizadas para dicho entrenamiento con `modelVariables`.

3.6. Recomendación sobre datos

En esta sección entraremos a utilizar el modelo generado por el entrenamiento con los datos encontrados en el repositorio abierto para clasificar los datos iniciales que ya disponíamos. En al figura 3.51 vemos el flujo en Taverna del programa **recommenderSystem** que nos asiste en tal tarea.

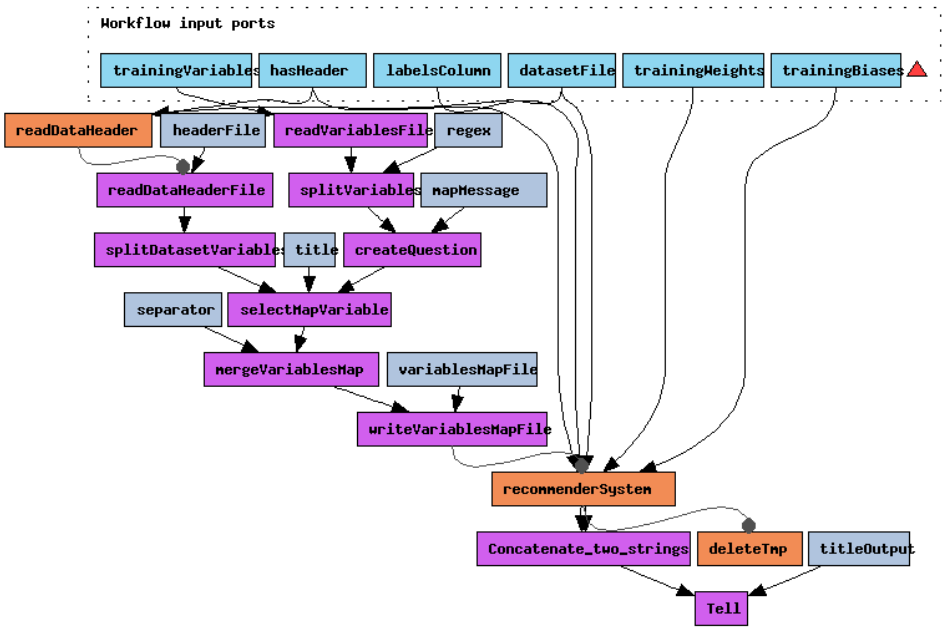


Figura 3.51: Diagrama con el flujo del proceso *recommenderSysmte* en Taverna.

Este proceso tiene un número mayor de puertos de entrada que hacen que su ejecución sea altamente configurable. Estas entradas son:

- **trainingVariables:** variable relativa al modelo recomendador a utilizar, esta debe ser la ruta al fichero de variables con las que se entrenó dicho modelo.
- **trainingWeights:** variable relativa al modelo recomendador a utilizar, en este caso debe ser la ruta al fichero con los pesos aprendidos del modelo.
- **trainingBiases:** variable relativa al modelo recomendador a utilizar, en este caso debe ser la ruta al fichero de parámetros *bias* aprendidos.
- **datasetFile:** ruta al fichero que contiene las muestras sobre las que se van a lanzar las predicciones.
- **hasHeader:** si el fichero a predecir contiene una línea de cabecera, se indica con el valor `--header`. Sino se deja con un valor vacío.
- **labelsColumn:** si el fichero a predecir tiene ya una columna de etiquetas, se debe indicar su nombre con el valor `--labelColumn COLUMNA`. Sino se deja con un valor vacío.

Continuando con el caso de uso, vamos a procesar nuestros datos iniciales con el programa *dataExploration* para reemplazar los valores perdidos por la media y luego realizarle una normalización por *feature scaling*. Del total de 699 muestras extraeremos 16 muestras concretas: aquellas que tuvieron valores perdidos. Este fichero lo llamamos `initial data processed.csv`.

El motivo de usar estas muestras es que el conjunto `replication data wisconsin.tab` descargado del repositorio, utilizado para entrenar el modelo, es un subconjunto de los datos de UCI que hemos usado como datos iniciales. Sin embargo, este conjunto obtenido con el programa *repoRead* no contiene ninguna de estas 16 muestras. Por ello, vamos a usarlas para que el sistema *recommenderSystem* las etiquete, pues el modelo entrenado no ha visto estos datos durante su aprendizaje.

Así pues, lanzamos su ejecución indicando: el fichero `initial data processed.csv` como *datasetFile*, el parámetro `--header` para *hasHeader* pues el programa de preprocesamiento *dataExploration* nos ha añadido una cabecera numérica, el valor `10` para *labelsColumn* e introducimos las rutas a los ficheros del modelo para los otros tres puertos de entrada.

Al arrancar el programa nos pedirá que vayamos indicando para cada variable del entrenamiento cuál es su correspondiente atributo/columna en el conjunto de entrada. Este paso es necesario para poder hacer corresponder correctamente los parámetros del modelo y las características de las muestras en los cálculos de las recomendaciones. Un ejemplo de este paso se ve en la figura 3.52.

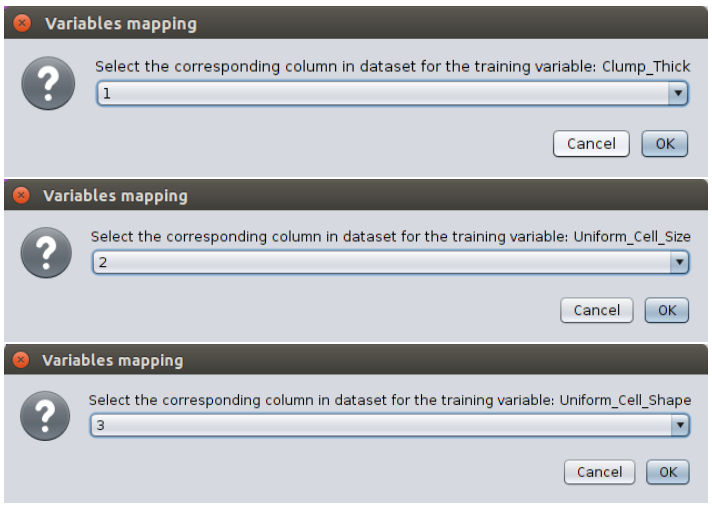


Figura 3.52: Ejemplo de mapeo de las variables del conjunto de entrada y las utilizadas en el entrenamiento del modelo reutilizado.

El programa permite que el conjunto a etiquetar posea más atributos que variables se usaron en el entrenamiento. En este caso, las columnas no mapeadas se borran del conjunto para poder trabajar con el resto.

Con todos estos pasos previos realizados, el sistema construirá un grafo de Tensorflow donde pondremos en varias variables los pesos θ, los valores de *bias* y el conjunto de entrada. Las predicciones consistirán en la aplicación del cálculo presentado en la ecuación (3.3) por lo que esto tomará un tiempo proporcional al número de muestras que se disponga. Al final, como se ve en la figura 3.53, el sistema indicará el porcentaje de acierto que ha tenido si el conjunto estaba ya etiquetado.

En la última línea el programa informa de que crea un nuevo fichero, en una ruta definida dentro del programa de Python que ejecuta Taverna, con el nombre del conjunto indicado como *datasetFile* más el sufijo `.predicted` en el que ha creado una nueva columna llamada `predictions` que incorporan las predicciones del sistema. En la figura 3.54 se puede ver este fichero para nuestro conjunto inicial.

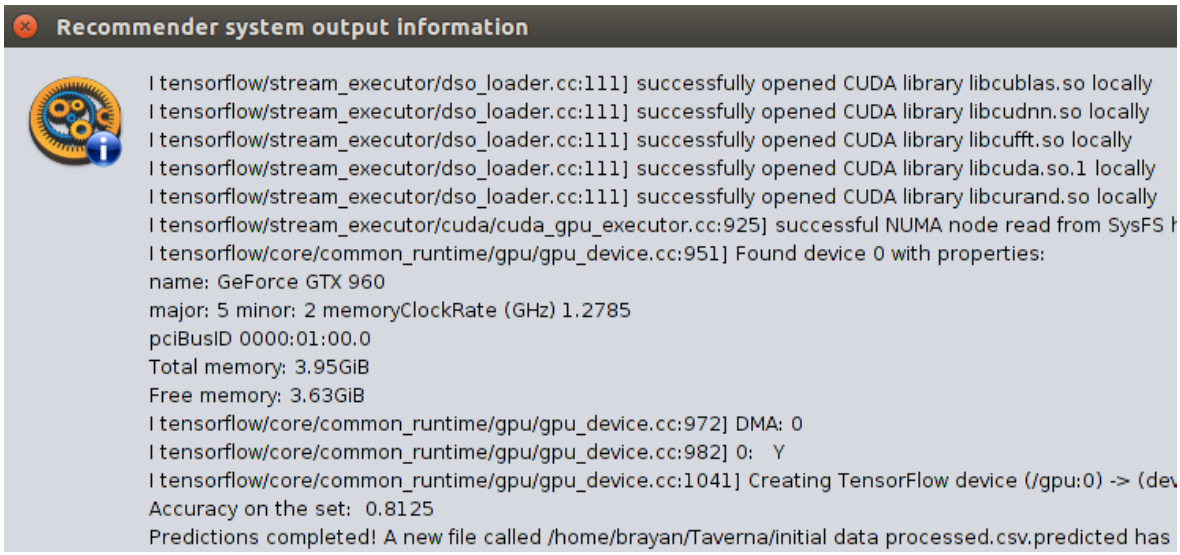


Figura 3.53: Muestra de mensaje con el que *recommenderSystem* finaliza su ejecución.

```
brayan@orionux:~/Taverna$ head initial\ data\ processed.csv.predicted
1,2,3,4,5,6,7,8,9,10,prediction
0.777777777778,0.333333333333,0.444444444444,0.0,0.111111111111,0.282739547747,0
.666666666667,0.222222222222,0.0,1.0,0
0.555555555556,0.555555555556,0.555555555556,0.888888888889,0.555555555556,0.282
739547747,0.666666666667,0.777777777778,0.0,0.0,1
0.0,0.0,0.0,0.0,0.0,0.282739547747,0.111111111111,0.0,0.0,0.0,0
0.0,0.0,0.222222222222,0.0,0.111111111111,0.282739547747,0.111111111111,0.0,0.0,
0.0,0
0.0,0.0,0.111111111111,0.0,0.222222222222,0.282739547747,0.0,0.0,0.0,0.0,0
0.444444444444,0.0,0.0,0.0,0.111111111111,0.282739547747,0.222222222222,0.0,0.0,
0.0,0
0.222222222222,0.0,0.333333333333,0.0,0.111111111111,0.282739547747,0.2222222222
22,0.0,0.0,0.0,0
0.222222222222,0.0,0.0,0.0,0.111111111111,0.282739547747,0.222222222222,0.0,0.0,
0.0,0
0.222222222222,0.0,0.222222222222,0.0,0.111111111111,0.282739547747,0.1111111111
11,0.0,0.0,0.0,0
brayan@orionux:~/Taverna$
```

Figura 3.54: Fichero con las predicciones realizadas por *recommenderSystem*.

3.7. Ejecución

Llegados a este punto se ha descrito por completo la herramienta propuesta con sus funcionalidades y el flujo de procesos que la componen.. Para que cualquier usuario pueda utilizar esta herramienta tal y como hemos hecho nosotros necesitará cumplir con los siguientes requisitos técnicos en su equipo de ejecución:

- Python 2.7.6 con las librerías `numpy, pandas, matplotlib` instaladas
- Tensorflow 0.11.0 instalado en el equipo, ya sea en su instalación para GPU o CPU
- Taverna Workbench 2.5

Cumpliendo estos requisitos, solamente hay que descargar los *workflows* definidos en Taverna que constituyen nuestra propuesta. Para ello, se han dejado habilitados en un paquete de *myExperiment* llamado `Open research data reutilization tool` en la dirección `https://www.myexperiment.org/packs/721.html` que contiene los *workflows*. Una vez descargados, es necesario realizar unas modificaciones sobre los servicios *Tool* que incorporan código en Python pues tienen especificadas algunas rutas absolutas que deben adaptarse al equipo.

Para ello basta con dar click derecho sobre el servicio, pulsar sobre `Configure tool invocation...` y navegar hasta `Advanced > Strings` donde encontraremos el código de Python, como se ve en la figura 3.55. A continuación se listan los puntos a modificar por cada módulo.

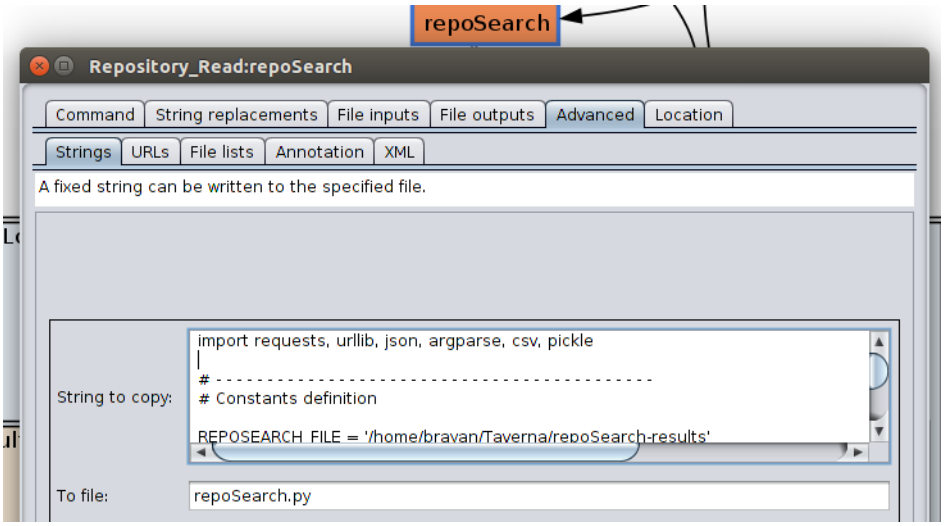


Figura 3.55: Ejemplo de modificación de *repoSearch*, del *workflow repoRead*.

repoRead

- **repoSearch:** cambiar la ruta `REPOSEARCH_FILE` por la deseada, también hay que cambiar la constante `DATAVERSE_API_KEY` para indicar la clave de autenticación en la API del Dataverse de Harvard. Si se explora otro Dataverse,

cambiar `SEARCH_API_POINT`.
- **resultsRead:** cambiar las ruta `REPOSEARCH_FILE` y `CURRENT_RESULT_FILE`.
- **resultsDownload:** cambiar las rutas `DATASETS_FOLDER` y `CURRENT_RESULT_FILE`, también tenemos los puntos de la API `DATAVERSE_API_KEY`, `DATASET_API_POINT` y `FILEACCESS_API_POINT`.
- **deleteTmp:** cambiar las rutas `REPOSEARCH_FILE` y `CURRENT_RESULT_FILE`.

dataExploration

- **loadDataset:** cambiar las rutas `TMP_DATASET_FILE` y `TMP_HEADER_FILE` por la deseada, estas se usarán en el resto del programa.
- **headerFile:** en el *workflow mainLoop* tenemos al inicio esta constante de texto, debe cambiarse por el valor dado a `TMP_HEADER_FILE`.
- **describeDataset:** cambiar la ruta `TMP_DATASET_FILE`.
- **subworkflows:** cada uno de los *workflows* dentro de *mainLoop* implementan las funcionalidades básicas del programa *dataExploration* con una *tool* que contiene un pequeños script de Python. Estos scripts contienen referencias a `TMP_DATASET_FILE` y `TMP_HEADER_FILE` que se deben cambiar.
- **deleteTmp:** cambiar las rutas `TMP_DATASET_FILE` y `TMP_HEADER_FILE`.

modelBuild

- **modelBuild:** tenemos que cambiar las rutas `LOGS_PATH`, `WEIGHTS_FILE`, `BIASES_FILE` y `VARIABLES_FILE`.

recommenderSystem

- **readDataHeader:** cambiar la ruta del fichero `TMP_HEADER`.
- **headerFile:** en *workflow* tenemos al inicio esta constante de texto, debe cambiar su valor por el dado a `TMP_HEADER`.
- **recommenderSystem:** cambiar la ruta del fichero `TMP_VARIABLES_MAP`.
- **variablesMapFile:** en *workflow* tenemos llegando al final esta constante de texto, debe cambiarse su valor para que coincida con el que se da a `TMP_VARIABLES_MAP`.
- **deleteTmp:** se deben cambiar las rutas `TMP_HEADER` y `TMP_VARIABLES_MAP`.

Capítulo 4

Conclusiones

Este capítulo discute los resultados obtenidos en el proyecto con los objetivos planteados al inicio. Además, se plantea trabajo futuro que pueda mejorar el sistema propuesto.

El trabajo de fin de máster documentado en esta memoria se inició con la idea de investigar en la reutilización de los datos de investigación en informática médica. El motivo de esta investigación venía de intereses personales en el mundo de los datos abiertos y el aprendizaje automático, así como la aplicación de estas herramientas en campos clínicos.

Con tal fin, al comprobar el estado actual de los datos abiertos de investigación, se propuso un sistema que pudiera dar uso de estos ya sea en el ámbito médico como en otros. Por ello, tras ver que los datos abiertos de investigación están siendo potenciados por normas internacionales y programas como el H2020 de la Unión Europea, decidimos proponer un sistema capaz de alimentarse de datos abiertos encontrados en repositorios ya que está claro que en un futuro muchos proyectos serán dirigidos por datos abiertos.

Se decidió construir un recomendador motivados por las necesidades identificadas por la Comisión Europea. Hoy en día la inteligencia artificial está experimentando una gran evolución y su uso es cada vez más frecuente. Por ello, construir un sistema que pudiera utilizar datos abiertos para aprender de ellos era un objetivo claro de este proyecto.

No obstante, la propuesta más innovadora que se ha querido realizar en este trabajo no es solamente reutilizar datos abiertos sino poder reutilizar el modelo de aprendizaje entrenado. La idea detrás de esto viene desde nuestra propia experiencia en la que nos hemos encontrado en proyectos de minería de datos donde es necesario extraer el conocimiento para etiquetar nuevas muestras, pero el conjunto de entrenamiento que se posee es extremadamente pequeño o no siempre se dispone de una muestra equilibrada ideal para entrenar.

Por todo esto, esta propuesta pretende ser una herramienta de uso para investigadores que se encuentren en este tipo de casos con unos datos (ya sean privados o públicos) que quieren tratar pero no pueden por su magnitud o por las características de sus categorías.

La herramienta construida permite a estos investigadores realizar búsquedas en un repositorio de datos abiertos como es el proyecto de Dataverse de Harvard. Con ello, pueden buscar más datos relacionados con los que ya poseen. Esta búsqueda no pretende ser un sustitutivo de la lectura de bibliografía, paso básico en investigación. Lo que pretendemos es que se pueda encontrar información sobre resultados y conclusiones sobre el tratamiento de datos junto con los mismos datos utilizados.

En cuanto al segundo objetivo planteado, hemos creado en este ecosistema de programas una aplicación que permite preprocesar y explorar conjuntos de datos con una funcionalidad básica. Esta es una funcionalidad de gran ayuda en este tipo de proyectos pues el investigador normalmente realiza una visualización de los datos para ver si faltan valores, si hay características con mayores escalas que otras, para ver qué proporción de clases hay en las etiquetas, etc.

Por otro lado, nos hemos valido de la librería Tensorflow liberada por Google para las aplicaciones del sistema capaces de cumplir con los objetivos de construir un modelo de aprendizaje y poder reutilizarlo. Gracias a dicha herramienta, la implementación realizada del modelo con la que cuenta el sistema propuesto es capaz de aceptar datos de distintas magnitudes en número de muestras como de variables. Además, la aplicación aprovecha al máximo la potencia del equipo donde se ejecute el programa consiguiendo una aplicación rápida y eficiente.

Para facilitar el uso y la reutilización de la herramienta propuesta hemos utilizado Taverna que nos ha proporcionado lo necesario para integrar los cuatro módulos que componen la herramienta en sencillos flujos de trabajo. Estos flujos se han definido de forma que todo lo necesario para ejecutar la propuesta estén dentro de los mismos y se han liberado a la comunidad a través del sitio *myExperience*.

Por estas razones, creemos que hemos cumplido con éxito nuestro principal objetivo de definir un proceso que permita reutilizar datos abiertos para construir un modelo recomendador que además puede ser reutilizado. Como resultado hemos obtenido una herramienta abierta y reutilizable que ejecuta este proceso.

En cuanto a trabajo futuro, el proyecto deja mucho campo que recorrer en cada módulo del sistema. Se plantean trabajos en la exploración de repositorios para considerar más de uno. Además, podría mejorarse la valoración de los resultados dada una cabecera utilizando técnicas de catalogado de librerías y procesado de lenguaje natural. También se encuentra interesante poder mejorar la búsqueda de forma que el sistema pueda reconocer que en un conjunto publicado se tratan campos similares a los datos que tiene disponible el investigador utilizando meta-características.

Para la exploración y *profiling* de los datos descargados, sería interesante añadir más tipos de visualización de datos y funciones de preprocesado como pueden ser técnicas de *downsampling*, *oversampling* o *Principal Component Analysis*. Sería de gran utilidad poder automatizar algunas de estas tareas de forma que el programa reconozca la necesidad de procesar el conjunto de forma autónoma.

Se podría investigar en el futuro en una forma de mejorar los aciertos del modelo de aprendizaje, ya sea utilizando otro modelo o dando la posibilidad al usuario de configurar uno entre una lista al estilo de otras herramientas de minería como Weka. Para la reutilización del modelo sería interesante trabajar en cómo poder utilizar un modelo entrenado con un conjunto de variables X y lanzar recomendaciones sobre unas muestras con un conjunto de variables Y, de tal forma que los conjuntos X e Y coincidan en algunas variables pero no en todas.

Nos gustaría dar pié con este proyecto a la creación de un repositorio de modelos abiertos creados con el uso de esta herramienta. Sería muy interesante liberar los resultados del módulo *modelBuild* para que otros investigadores puedan reutili-

zar esos modelos sobre sus datos, de forma que ya no necesitarían entrenar ellos el modelo ni mucho menos buscar otros datos para crearlo.

A modo de conclusión nos gustaría motivar a la comunidad investigadora para que empiece a trabajar con una cultura de datos abiertos de manera que podamos recuperar el espíritu científico de compartir conocimiento liberando los datos por el bien común.

Bibliografía

[1] Fernanda Peset. ¿Qué hacen los investigadores con los datos de los proyectos? En EDAUA15, http://www.slideshare.net/uadatos/fernanada-peset-qu-hacen-los-investigadores-con-sus-datos, 2015. Acceso: 09-01-2017.

[2] Líderos del G8. G8 open data charter and technical annex. https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex, 2013. Acceso: 11-01-2017.

[3] Comisión Europea. Hacia una economía de datos próspera. http://ec.europa.eu/transparency/regdoc/rep/1/2014/ES/1-2014-442-ES-F1-1.Pdf, 2014. Acceso: 11-01-2017.

[4] Comisión Europea. Guidelines on open access to scientific publications and research data in horizon 2020. http://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/hi/oa_pilot/h2020-hi-oa-pilot-guide_en.pdf. Acceso: 11-01-2017.

[5] Open research data pilot - openaire. https://www.openaire.eu/. Acceso: 11-01-2017.

[6] Zenodo. https://zenodo.org/. Acceso: 11-01-2017.

[7] Datacite. https://www.datacite.org/. Acceso: 11-01-2017.

[8] DataCite. r3data - registro of research data repositories. http://www.re3data.org/. Acceso: 11-01-2017.

[9] Figshare. Figshare. https://figshare.com/. Acceso: 11-01-2017.

[10] Dryad. Dryad. http://datadryad.org/. Acceso: 12-01-2017.

[11] Open Knowledge Foundation. Datahub. https://datahub.io/. Acceso: 12-01-2017.

[12] Harvard. Dataverse. http://dataverse.org/. Acceso: 12-01-2017.

[13] MyExperiment. Myexperiment. http://www.myexperiment.org/. Acceso: 12-01-2017.

[14] ResearchObject. Researchobject. http://www.researchobject.org/. Acceso: 12-01-2017.

[15] Opening Science. Software and tools. http://www.openingscience.org/category/initiatives/software tools/. Acceso: 12-01-2017.

[16] Python. Python logo. https://www.python.org/community/logos/. Acceso: 09-01-2017.

[17] Dataverse Project. Dataverse logo. http://dataverse.org/. Acceso: 09-01-2017.

[18] Dataverse Project. Guía api. http://guides.dataverse.org/en/latest/api/index.html. Acceso: 09-01-2017.

[19] Google. Tensorflow logo. https://googleblog.blogspot.com.es/2015/11/tensorflow-smarter-machine-learning-for.html. Acceso: 09-01-2017.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[21] Google. Ejemplo de grafo en tensorboard. https://www.tensorflow.org/how_tos/graph_viz/. Acceso: 09-01-2017.

[22] Apache. Taverna logo. https://svn.apache.org/repos/infra/websites/production/taverna/content/ Acceso: 18-01-2017.

[23] Apache. Ejemplo de workflow en taverna. https://taverna.incubator.apache.org/introduction/why-use-workflows. Acceso: 18-01-2017.

[24] UCI. Breast cancer wisconsin (diagnostic) data set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic). Acceso: 10-01-2017.